DEPLOYABLE ARCHITECTURE

A Masters Thesis
Presented to
The Academic Faculty

by

Andre M. James

In Partial Fulfillment
of the Requirements for the Degree
Master of Architecture in the
College of Architecture at Georgia Institute of Technology

Georgia Institute of Technology
Summer 2008

DEPLOYABLE ARCHITECTURE

Approved by:

Lars Spuybroek, Advisor
College of Architecture
Georgia Institute of Technology

Kevin Young, AIA
Georgia Institute of Technology

Dr. Gernot Reither
College of Architecture
Georgia Institute of Technology

Ellen Dunham-Jones
College of Architecture
Georgia Institute of Technology

Date Approved: April 1st, 2008

ACKNOWLEDGEMENTS

To my family who has been my closest friends and my friends who have been as close as family, and my best friend that I lost too soon: Thank you! You gave me strength!

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

SUMMARY

Folding empowers the user to change the form and function of a sheet of paper through a sequence of manipulations. Unfolding the once folded artefact produces a diagram that describes its own making that can be replicated at different scales using a new material. Architecturally, folding can be employed a morphogenetic solution to design a system that can be fabricated from a sheet material, that like paper, can be folded into a inhabitable structure.

The ease and cost efficiency of fabrication based on folding can be used to design a system that executed using low cost materials can be used as a shelter that accommodates programmatic and aesthetic evolution. Thus, the system lends itself to being a transitional shelter for communities that have been displaced due to a natural disaster or other form of crisis.

Technological advances in design and structural analysis can give the designer the power to define the complex process folding parametrically allowing the input a real-time feedback based design based on an a folding inspired algorithm.

CHAPTER 1: INTRODUCTION

Thesis Question

How can today's progressions in technology architecture, namely parametric design, morpho-ecology/genesis and emergence, machining and its result prefabrication construction system/s, be used to design rapidly deployable assembly ready *architecture*?

Context and Program

Hurricane Katrina in New Orleans, Louisiana, the earthquake in Lima, Peru, and a suicide bombing in Amerli, Iraq, all left their respective geographic locations in ruins, pushing the survivors into temporary shelters while they are rebuilt. By implementing a prefabricated structure we can regenerate tragedy struck areas by replacing the temporary tent shelters that are used with a system that accommodates growth and permanence. Designing an intelligent system founded on a easily comprehensible logic that can be assembled with little effort, skill and infrastructure can stimulate the transformation of the refugee camp to community.

Through the analysis of both the mechanical and conceptual techniques of folding, bending and creasing found in origami and studying their inherent structural properties for 1) the formulation of a surface skin system and 2) the ability to generate various types of spaces, a deployable system of architecture will be developed to capitalize on the economies of production of using a sheet material. The congenital characteristic of folding that allows the material to be compressed along the axis of the fold becomes integral to the concept of easily transporting the system with a high level of spatial efficiency. This, thus, lends itself to nesting, compressing or further folding of the system in its "dormant" phases for transportation. Activation of the system for assembly would entail unfolding the system and the simple assembly of components by an individual or group depending on the scale and complexity of the construction type.

The goal of this study is to adapt principles gained from studying scaled manipulations of paper to full scale prefabricated components that perform as well as,

or better than their scaled counterparts. Analog investigations at various scales of the transfiguration from sheet to geometric would be the preliminary means of research. Modeling between scales will cover the structural limits of the paper and its inability to span without the introduction of additional ridges. Introducing paper thin materials such as plastics, and/or metals introduces new performative attributes and limits to the folded system. Once analog surface skin models are devised, the system can become a component, allowing it to circumscribe programmatic configurations of space and and evaluate its performance as a complete system. Devising a system of components as opposed to a kit of parts, allows a range of use and configuration that is based on performance in lieu of aggregation. The components could be rationalized into multiple pieces to be cut from a sheet, transported to the site in its dormant phase and activated for inhabitation.

CHAPTER 2: SHELTER AND ARCHITECTURE

Architecture's Origin in the Shelter

Living organisms have natural habitats in which they grow and thrive; in some instances they alter their surrounding in order to produce their home, while other creatures leave the environment unchanged. Man sets himself apart from other animals because of his ability to adapt wide-ranging natural conditions and materials to provide shelter from the elements. However, his manipulation of his surroundings has evolved to address, comfort, security and ultimately stability as his nomadic habits changed. Laugier used the notion of the primitive hut to illustrate architectural discourse (*Figure 1*): though the cave provided the shelter from the sun, rain and elements, it isolated its inhabitant encapsulating him in the dank, uncirculated air, isolating him from the world. As a result, he constructed the first house with branches for structure and leaves for enclosure as a solution to the problem of shelter without isolation.[1]

Figure 1: The Primitive Hut. Anthony Laugier's depiction of the first instance of architecture.

Vitruvius offered that the birth of architecture was in the discovery of fire that lead to the congregation of individuals around the warmth (hearth) of the fire. This gathering

1 *Laugier, Marc-Antoine. Essay on Architecture 1753*

of people stimulated conversation and intellectual growth; man innovated caves and constructions seen in nature to his own means, building shelters that over time evolved to improve the performance of the home. Their teachability allowed them to improve on their knowledge; reflecting on their own discourse, therefore generating a feedback loop.

Shelter provides the basic means of protection from the elements and though their assemblies and construction may be crude or refined, the life span of the shelter under the continuous use of human inhabitation is, as the word 'shelter' insinuates, relatively short. According to the Office for the Coordination of Human Affairs, referred to as OCHA, a shelter is "a habitable covered living space" used to keep people safe and to help people retain their dignity in emergencies[2]. Tent structures are the most ubiquitous form of deployable shelters known possibly due to its well documented history of utilization. The tents distributed by the United Nations closely relates to Laugier's primitive hut, replacing the natural materials of branches and leaves with man-made fabrics. It is this utilization of varied material types combined with the ability to aggregate tents in multiple configurations in addition to providing near immediate shelter, that makes the tent the easier option, despite being deemed as a last resort and temporary solution. Tents can be assembled without the need of supplementary tools or specialized knowledge, which is another advantage for its implementation in crisis situations despite the simplicity of its design, which makes it difficult to adapt to permanent use. However, the priority in such situations is to provide shelter to the afflicted, not to provide a permanent solution. The procedure for providing shelter to displaced persons is not limited to the individual shelter but also to the deployment of the overall system.

The emergency strategies as outlined by OCHA work effectively to provide refuge to displaced persons and are designed to be temporary recourses to a better solution. The transitional shelter is meant as an intervention between allowances and features. Vitruvius' parable about the evolution of architecture as a result of discourse explains the notion of the transitional shelter. Over time, the crude shelter is recursively reconfigured to provide better amenities for living, adapting observations to generate innovations that improve the construction and its space. Restoring aesthetic considerations into the design phases of the transitional shelter provides a basis that potentially reintroduces a culturally contextualized adaptation of its architecture in the development of the new community from the refugee camp.

2 *Tents. A Guide to the Use and Logistics of Family Tents in Humanitarian Relief. United Nations.*

The Transitional Shelter Strategy

The United Nations has devised strategies to serve the lives and welfare of persons that are afflicted by crises that warrant the unanticipated, mass displacement of people. In many scenarios, the utilization of encampments is the most feasible recourse to alleviate the current situation; conversely, not all shelters are tents. Unfortunately, this solution is not designed for societal longevity. Despite allowing the displaced to make minor alterations to their tent to improve their comfort in ways they see fit, the tent doesn't facilitate permanence without introducing new construction is not designed for such parameters. Ultimately, the living conditions of a refugee camp (*Figure 2)*, in a majority of cases, never improve and sometimes even deteriorate over time. Ideally, the encampment is meant to provide shelter until the people who have been displaced are able to return to their former community or are absorbed by another. However despite being designed as a temporary solution, the formed refugee camps are at least semi-permanent, outliving the approximate 18 month lifespan of the materials used to make the tents in which displaced reside.

Figure 2: A Sudanese Refugee Camp

Since different peoples utilize different methods of construction based on their culture and accessible materials it is possible to introduce an infrastructure that allows immediate inhabitation that can be built upon and re-appropriated as the intensity of the crisis dissipates, the shelter evolves with societal stability. Climate, culture and needs influence the development of the base activating the innate architecture through contextual interaction. Through its adaptability, the transitional shelter is meant to be phasing system that begins to provide more than the minimum shelter requirements while

permitting a versatility of improving upon the system.

In principle, excogitating the master plan (*Figure 32*) of the transitional encampment is no different than designing a small town or village. Redeveloping in the wake of crisis requires careful consideration in utilizing the available or projected resources and infrastructure; communalized resources become a necessary strategy of conservation. While arranging the shelters in clusters simulates the natural arrangement of homes in communities, coinciding their placement with the location of food, water sources or other important infrastructure. Strategizing multiple points for centralized resources allows for the intelligent rationing while providing allowances for growth and development (. Over time, as the existing assets improve their related cores stabilizes and begins to develop. Furthermore, introducing more nuclei accommodates the overall growth of the development overlapping and ultimately eliminates the bounds of each nucleus.

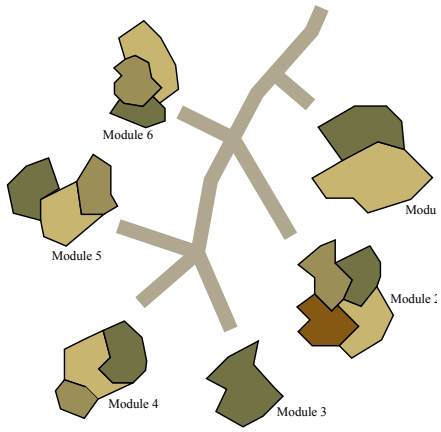

Figure 3: Diagram of potential module clustering to form sub-communities

Employing constellation as a nested strategy produces a greater density of persons per nucleus which proves beneficial in scenarios where the resources may not be as clustering multiple families into each unit of the local sub-group. As such, despite the initial design intention of the shelter to function as a single family unit, the design allows a reconfiguration of program, without an extensive reconfiguration of structure or skin. Hence, the initial implementation permits multiple families to occupy the space to increase the distribution of shelter and resources per core. For instance, two families can work in conjunction to assemble the unit in which they will live. When another unit needs to be assembled, the secondary household occupying the aforementioned shelter

will cooperate with a secondary family from another household to assemble another unit and upon completion, both households would move into the newly completed unit (*Figure 4*). The process would be repeated as necessary to complete the required units; with experience each household is able to understand the principles of the system which would assist in the future evolution of their home. Over time, the number of single family within the community will increase. Similarly, to cater for the juncture of the individual household, the infrastructure can be used in reverse for individual households, allowing spaces to be re-programmed to accommodate second household to support the primary household.

Figure 4: Example of programmatic reconfiguration

Both the programmatic reconfiguration and adaptation and the physical alteration of the structure, generate a feedback loop which influences the development and the architecture inherited from the concept of the system. Deductively, designing a parametric solution to the transitional shelter allows an accommodation of scales of crises enabling the inhabitants to adjust their program easily without changing the configuration, or by modifying the configuration without changing the underlying infrastructure.

CHAPTER 3: FOLDING IN ARCHITECTURE

The History of Folding in Architecture

Chuck Hoberman refers to the unfolding of architecture as "an object that is identically a structure and a mechanism[3]", boasting that such an innovation allows a controlled transfer of forces and motion within the system without the need for any secondary support systems. He continued to say:

> Underlying these unities of structure/mechanism and fluidity/strength are unique mathematical principles. The elegance and economy exhibited by unfolding architecture derive from this mathematical and geometric basis. The basis of each folding structural system is embodied in a minimum number of representative connected parts... Unfolding structures are made up of simple part with simple connection between them.

He posits that surface structures, made by repetitive pleating from a single surface or sheet, can transform smoothly between an extended structural configuration (active state) and a compact bundle (dormancy). Surface structures in turn act like hinged rigid plates allowing fluid, kinetic behaviors since the system acts as a mechanism defined by the matrix of folds. Folding has been a tool for the morphogenetic exploration of program and space and to a lesser extent, an investigation of a structural skin. For the purposes of rapid deployment and assembly, constructions based on the mechanic and geometric principles applied to a sheet surface are especially economically prudent. Not only does the folding of the material affect the structural integrity of the system but so does the thickness of the sheet itself.

Transferring this diagram to another planar material gives the designer the chance to impose the logic of the folds onto the new material. The polygons defined by the creases of the paper can be fabricated with a non-pliable material, with the creases being substituted for mechanical hinges or static connections. Material characteristics (pliability, thickness for example) may restrict the literal translation of the folding from being conveyed, the performance of that manipulation becomes architectonically

3 *Hoberman, C. (2004). "Unfolding Architecture." Architectural Design*

manifested through the details of representation and fabrication.

Tracing the outline of the two-dimensional shapes, connecting their vertices with linear members produces a three-dimensional truss system. Moreover, if the members were able to rotate around the vertex, one would be able to develop a foldable truss system that can expand and retract in the same fashion as a folded surface. Fabrication of a folded truss system benefits from directly translated to a truss system members on rotational hinges at vertex points. The chronology (*See the Fishbone Pleat Algorithm*) of manipulations are crucial to the development of the artifact making the folding an event as much as it is a technique. Changing the sequence of manipulations sequence change the geometry of the final artifact, in some cases restricting the type or number of manipulations can be carried out without the paper having to yield unnaturally. Unfolding a succession of folding manipulations renders diagrams that can be used as instructions to inform its own making (*Figure 5)*.

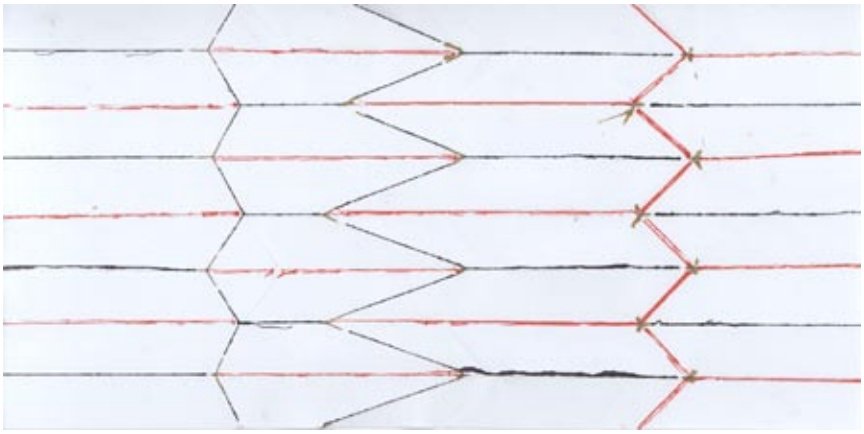

Figure 5:
Unfolded sheet from Fishbone Pattern
Red Lines – Valley Creases. Black Lines – Mountain Creases

Implementation of a repetitive folding pattern permits expansion by performing the same transformation of the span of repetition. In conclusion, defining a partition using a pleated pattern results in the repetition of a profile or cross-section; this principle is applicable to both folded surface and folded truss systems, meaning corresponding components of such a system remains invariant despite its spatial translation.

Folded Architecture Precedent:

The Yokohama International Port Terminal[4].

Yokohama, Japan. Foreign Office Architects

> "Our proposal for the project start by declaring the site as an open public space and proposes to have the roof of the building as an open plaza, continuous with the surface of Yamashita Park as well as Akaranega Park. The project is then generated from a circulation diagram that aspires to eliminate the linear structure characteristic of piers, and the directionality of the circulation."
>
> -FOA*

FOA employed a folded surface strategy to accommodate a large interior span in addition to the reconciliation of the seismic forces that affect Japan. The self-triangulation of the implemented folding technique informs the faceted surfaces which are connected to produce a static architectural element. These triangular faces distribute the structural loads diagonally towards the ground (*Figure 7*). The folding strategy also permits visual continuities along the faces of the structure for both the interior and exterior areas. To augment the perception of continuity, FOA applied similar finishes along the faces of the building, only varying the materiality of the surface to denote changes in the architectural program (*Figures 8-10*).

Figure 6: Example of Folding Tesselation using Standard Copy paper. Richard Sweeney

4 *ARCspace.com. < http://www.arcspace.com/architects/foreign_office/yokohama/yokohama_index.htm>*

Figure 7: Interior Photograph of Folded Roof Structure.

Figure 8: Photograph highlighting the binary conditions of the folded roof structure.

Figure 9: Interior photograph exhibiting the surficially informed finishes.

Figure 10: Exterior photographs exhibiting the surficially informed finishes

Folding as a Design Algorithm

The folding rule set is defined by the manipulations that could be carried out on a sheet of paper to elicit change. Sophia Vyzoviti describes these manipulations as: folding, pleating, creasing pressing scoring, cutting, pulling up-down, rotating and twist. Turning, wrapping enfolding piercing, hinging, knotting, weaving, compressing, balancing and unfolding complete the list of manipulations; all of the aforementioned can be used in conjunction with each other or individually to generate a developable object[5]. Of those manipulations, the fold, crease and the pleat were chosen as the primary operations to be used for the development of the RDA system, utilizing a 1:2, length-to-width sheet material; the ratio is set by the standard dimensions of a sheet of plywood. These three manipulations could be adapted to perform as the hinge and allow compression; unfolding produces its fabrication diagram.

The "fishbone" pattern (*Figure 11*) is based on a pleated algorithm that is acquired by reversing the cross-sectional folds along the pleats, alternating the plane of the crease, between mountain and valley (*Figures 12 & 13*), of the pleat at each fold's vertex. The corrugation from the pleating increases the structural capacity of the paper allowing it to span distances, performing as a series of one-way joists. Akin to the joist, the depth of the crease, which can be tweaked by adjusting the frequency of folds, increases the structural capacity of the system. However, accomplishing larger spans may require the thickness of the material to increase. The fishbone pattern proves useful to this application of folding in architecture because each pleat remains co-planar in normal operation ranges.

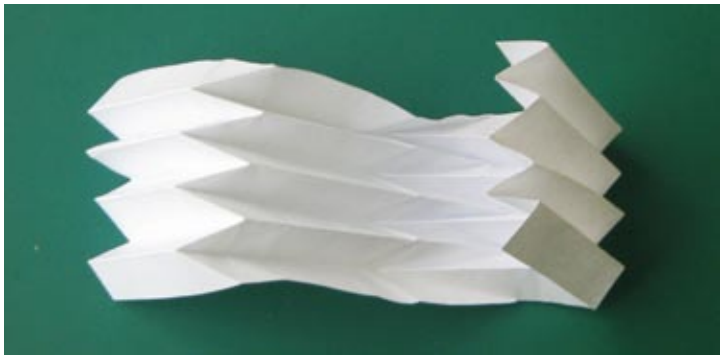

Figure 11: The Fishbone pleat executed using Standard Copy Paper

5 *Vyzoviti, Sophia. "Folding in Architecture. Spatial, Structural and Organization Diagrams."*

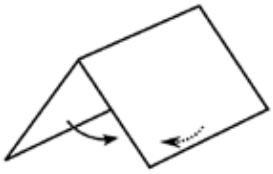

Figure 12: Mountain Fold

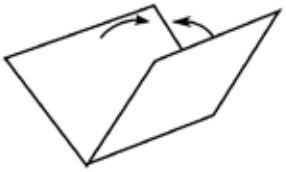

Figure 13: Valley Fold

The Fishbone Pleat Algorithm

1. Fold sheet lengthwise and crease the edge.
2. Fold the sheet in half again, creasing the edge, repeating this step as many times as desired or physically possible with the material.
3. Unfold the paper and refold along established creases to make a basic pleated fold (*Figure 14*).
4. Compress the pleated paper and make a crosswise, diagonal crease.
5. Unfold paper to reveal the creases made from the previous operation. Reverse the direction of the folds to on one side making an 'outside reverse' (*Figure 15*) or 'inside reverse' (*Figure 16*) fold depending of desired material re-direction.
    - 5.1 The Outside Reverse Fold wraps the wings on one side of the crosswise fold around the crease of the fold.
    - 5.2 The Inside Reverse Fold tucks the wings on one side of the crosswise fold inside the wings of the other side.
6. Compress the new fold to set creases.
7. Repeat steps 5 and 6 as desired.

Figure 14: Pleat Folds

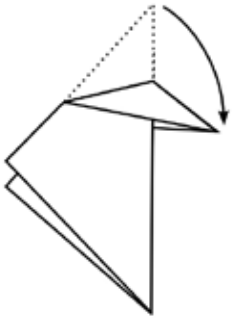

Figure 15: Outside Reverse Fold

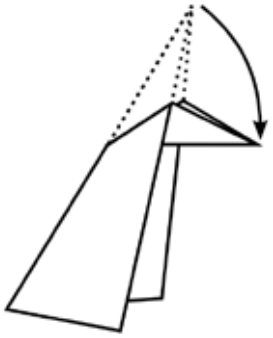

Figure 16: Inside Reverse Fold

The Fishbone Algorithm for the Generation of Connected Rigid Plates

1. Divide sheet lengthwise into desired number of equal parts.
2. Define quadrilaterals that are to be extracted from each length of material.
3. Cut each length into multiple parts to produce facets by cutting across the length as defined by the creases in a folder version of the structure.
4. Repeat previous step for each length of material.
5. Arrange the trapezoids as informed by fishbone pattern. i.e. Inferring that the parallel sides of the trapezoid defines the y-axis, start at the bottom-left corner, place the next trapezoid of the column by lining up their matching adjacent edges until all the trapezoids of the first column are set.
6. The adjacent columns simply reflect about the edge parallel to the y-axis until all the quadrilaterals are placed to form the complete sheet.
7. Connect each trapezoid across their four adjacent edges, thusly, each tetragon connects to 4 other shapes.

The reverse fold in the fishbone pleat system simulates the intersection and trimming of two ‘ribs’ that are rotated about an axis that is perpendicular to the ridge. The resultant edge delineates the plane of symmetry of the fold, thus, a line perpendicular to the plane passing through the vertex of correlating creases equally bisects the folded angle. In other words, a V-shaped cross-section of one of the ‘legs’ that extend from the reverse fold reflected by the mirror plane defines the cross section of the other side of the reverse fold (*Figure 17*). The size of the angle along the surface of the pleat, defined by the crease lines that connect the upper and lower ridges) is inversely proportional to the angle at which the pleats are folded. The fold produced from creasing the compressed

pleats as in Step four (4) of the fishbone algorithm paired with the amplitude of the pleats inform the change in direction of the material on the opposite side of the crease.

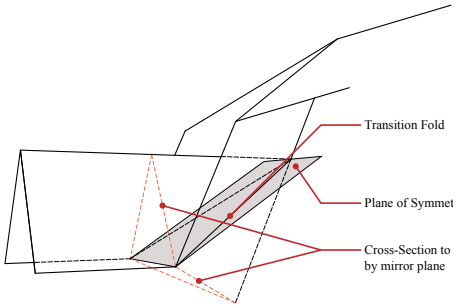

Figure 17: Anatomy of a virtual reverse fold

Once unfolded, the fishbone pleat is shown to comprise of a series of trapeziums (*Figure 18*) reflected along the ridge of each pleat. The geometries of each trapezoid can be reproduced from sheets of rigid material and connected by hinges to replicate the folded geometry based on a hinged rigid plate system. The inherent repetition of pleated surface is analogous to an extruded cross section which can be redefined by changing the cross section of the object along its length. Each quadrilateral represents a swappable facet system that can now be substituted to for simple modular repairs our augmentations. Incidentally, employing a system based on assembly modularity allows synchronous fabrication of the subgroups required to complete the structure. Furthermore, teams can assemble modules and assemble them to complete the house; likewise, sub-groups may be selectively omitted to address anomalies.

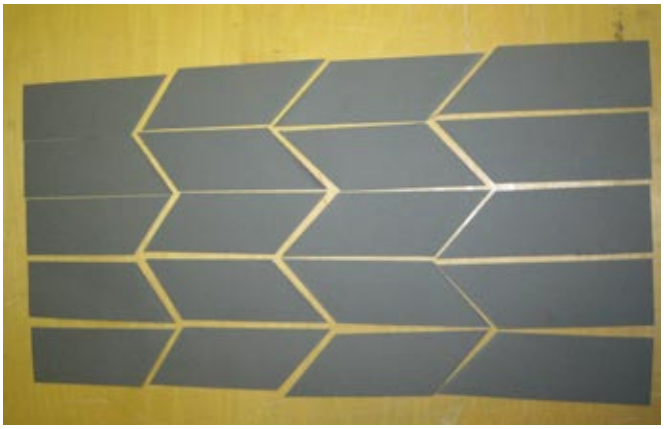

Figure 18: Series of trapeziums generated by fishbone pleat algorithm

CHAPTER 4: MATERIALS

Paper

Origami, by definition is the art of paper-folding; as such it is the basis of comparison for the folding manipulations to be carried out. As cardboard is simply densely laminated paper, its structural capacity is a direct relation to the number density of layers. Shigeru Ban used cardboard tubes as structural components in several constructions (*Figures 19 & 20*) and after conducting several tests has provided the following structural capacity information[6] (Table 1). Moreover, thicknesses of an ⅛ on inch or more, high-density cardboard could be easily be manipulated with wood working tools. These assets make a paper based product such as cardboard a logical choice to explore since it can be investigated at multiple scales and feasibly practical at full scale.

Figure 19: Paper Dome. Masuda, Gifu, Japan.

Figure 20: Paper House. Lake Yamanaka, Yamanashi, Japan.

Outer Diameter (mm) = 291			Inner Diameter (mm) = 250		
Specimens	Max. Compressive Strength (kgf/cm^2)	Young's Modulus ($x\ 10^3\ kgf/cm^2$)	Poisson's Ratio	Water Content in Percentage of Total Weight	Density (g/cm^3)
C - 1	97.9	21.8	0.134	10.2	0.816
C - 2	101.6	20.9	0.138	9.9	0.821
C - 3	101.4	21.4	0.140	9.9	0.819
C - 4	97.7	19.8	0.142	9.8	0.811
C - 5	97.7	21.6	0.151	10.2	0.817
Means	99.3	21.1	0.141	10.0	0.817

Table 1: Compressive tests results for cardboard tubes

The initial investigation utilizing paper as a material started with standard sheet

6 *McQuaid, Matilda. Shigeru Ban. New York: Phaidon, 2003*

of 20 lb copy paper 8½ inches x 11 inches and slightly heavier ivory carton[7]. The sheets were trimmed into smaller sheets measuring 4¼ inches x 8½ inches to conform to the 1:2 dimensional ratio of plywood[8]. The sheets of paper were folded using the steps outlined in the fishbone algorithm to generate 4 pleats, with one half flanking each side of 3 full pleats. Though reverse folds using copy paper has a slightly higher level of difficulty to execute than basic folds, the pliancy of the material allows it to be greatly contorted without marring the paper. In cases where the paper has been manipulated beyond its modulus of elasticity, the continuity of the paper fibers is broken resulting in wrinkling or creasing. The formed wrinkles identify the stresses within the material and are schisms of the force redistribution that is necessary to maintain the internal equilibrium of forces. In essence, wrinkling and creasing are the form-finding effects of paper-like materials. Under further examination, one would notice the degradation of the paper fibers that traverse the creases which degrade over usage until they break, splitting the material structure. Due to the excessive shear forces, the creases themselves are the weak links of folded structures. Unfolding the pleats reveals the trapezoidal creases along with the lesser imposed secondary creases and wrinkles that were formed as a result of the process of making of imperfections of the material. The heavier ivory carton displayed less secondary wrinkles and creases due to its greater rigidity afforded by cross-hatched organization of the fibers which also increased the durability of the creases.

Repeating the procedure with 54 lb bristol demonstrated a much greater difficulty to execute reverse folds at the same size due to the rigidity of the material. Accomplishing reverse folds imposed more prominent secondary creases in the material which affected the overall fluid compression and expansion of the pleats until the material has fully yielded under its stresses at the creases. To find the scale at which bristol could be utilized to replicate the structure that was generated using ivory carton, the dimensions of the starting sheet was scaled up by 25% per iteration until the reverse fold could be executed without introducing prominent creases. After 4 attempts the initial result was replicated using a sheet that measures 10⅜ inches x 20¾ inches. Another caveat bristol's rigidity gained from the increased material thickness is its inability to be accurately creased by folding the pleated sheet. Not only does the visibility of the creases diminish, from the innermost to the outermost layers of the fold, but the location of the crease is also offset by the approximate thickness of the material which prohibits the completion of the fishbone structure.

To compensate for the material thickness and the inability to crease by folding, supplementary means of scoring the material were investigated. Using a razor sharp

7 *From Dutch term: Ivoire Karton. A brand of linen paper that is sturdy, yet easy to cut.*

8 *See Chapter IV: Materials: Wood.*

edge or blunt pointed tip to gently cut the top surface of the paper allows folding in the opposite direction of the scored face. Scoring breaks the continuity of the fibers that cross the crease, and ultimately alleviates the amount of tension on the material in order to allow folding; better results are achieved from scoring with a sharp blade because the paper fibers are actually cut rather than displaced or deformed from using a blunt tip such as an ink-less ball point pen. However, scoring considerably weakens the material; in essence, the crease is merely a thinner section of the overall material and therefore accommodates better folding. Executing the fishbone pleat through scoring as opposed to folding as a means of inducing the creases requires careful consideration for the direction of the fold owing to the fact that the sheet must be scored on the opposing sheet face, with the score lines starting and ending precisely at vertex points. Capitalizing on the precision of a digital system, the laser cutter can be used to cut and prepare the fish bone folding diagram. The ability to set the speed and power level of the laser allows lines that are identified as creases to be scored consistently at their precise location, however, this method shares the same major disadvantage of scoring by hand which is the need to flip the sheet to score the other side to allow folds in the opposite direction. Generating creases in this manner, however, infringes on the bi-directionality of the fold since the fold can only be made away from the scored face.

Using a pounce wheel to crease the paper offers a better result than scoring with a blade or pen-point by perforating the sheet. Perforation keeps most of the fibers along the crease in tact, while also allowing the holes to alleviate excess stresses which can make the act of folding more difficult. The size of the pounce wheel and spacing of the teeth are defined by the weight or thickness of the material to be scored; a 3/16 inch diameter pounce wheel was used for papers rated with weights of 26 lbs or less, a ¼ inch diameter pounce wheel was used for the heavier grade bristol. Scoring by perforation eliminates the need to consider the sidedness of the material because the crease allows folding in both directions. Whereas the depth of scoring is controlled by the pressure applied and edge of the blade, which is inherently inexact, the spacing of material and void remains generally consistent as the characteristics of the perforation relies on the size of wheel and number of equi-spaced teeth on the wheel. Set by 2 constants, defining creases with the pounce wheel, though not exact, is more precise than scoring with a blade.

Folding matte board was not possible without heavily scoring the material with a sharp edge to compensate for the thickness. As a multi-ply material, matte board benefits from the laminating multiple layers of paper which increase the density of paper fibers per square inch. Although matte board can be bent without scoring, the imprecision of the crease makes it difficult to generate a clean fold; the tension in the material forces

the outermost fibers of the paper layers to yield. Albeit scoring allows the construction of the initial pleats, the overall rigidity of the material does not permit the maneuverability to perform the reverse folds. Despite its inability to be folded, matte board exemplifies a material that can be assembled according to the logic of paper-folding. The utilization of materials that lack the pliancy to be bent without secondary wrinkling are simply cut into multiple quadrilaterals generated by the logic of the fish bone system and reassembled as a series of connected rigid plates. Reconnecting the shapes with masking tape restores the axial rotation of the trapezoids about their edges which ultimately restores the dynamism of folding to the reassembled rigid plates. At this stage, the durability of the tape plays an important role in the reassembly of the matte board model.

Plastic: Styrene

Styrene is a rigid plastic that is used to make items such as plastic eating utensils and is available as uniform sheets of varying thickness that are used to fabricated sturdy plastic models. Though it is very flexible, it has a relatively low modulus of elasticity (λ = ±260,000 psi) and as such concentrating the force on a smaller area produces permanent deformation with a smaller amount of force. At this point the fibers of the material will degrade as more pressure is applied until complete failure. Accordingly, creasing styrene results in breaking along the proposed crease line even at smaller thickness of material and scoring only works for simple folds that do not go past the yield point as the material would fracture. Though it is possible to generate more complex folds with styrene, either direct heat is necessary to increase its pliability to fold it manually or with some form of press. As a plastic, styrene is also castable, however, utilizing molds to achieve the outcome is expensive and requires multiple molds to achieve the necessary configurations; the use of such advance tools, equipment and procedures limits its use as a means of production in the field. Thus, executing the fishbone pattern using 1/16 inch thick styrene sheets is only achievable by connecting the extracted trapezoids with rigid or hinged connections.

Steel

Steel (λ = 30,000,000 psi) is a plastic material, hence it can be processed similarly and, based on Hooke's Law, its performance can be analyzed and graphed and the final result will be very similar to plastic barring it's higher yield points. Because of its high

elastic modulus, steel can be bent to acute angles without breaking at the crease, however, repetitive stress, such as repeatedly bending the steel back and forth about the crease, will cause the steel to break as the “fibers” degrade with each manipulation. Consequently, accomplishing the pleat pattern, barring means which require special skills, tools or equipment, is done through connecting the cut-out quadrilaterals which also required an intermediate skill and tools set to manipulate the material in the field. This fact, combined with its cost, makes steel an impractical choice as the primary material for fabrication the RDA. However, steel could still be utilized for the fabrication of the connections between faces. Steel barrel, butt or piano hinges could be used to offer a folding dynamic whereas bending plates at predetermined angles offer strong rigid connections between adjacent faces of the structure.

Wood

Wood’s history as a construction material makes it a worthy selection for both scaled and full size fabrication. Wood is available in sheet form in the form of plywood, particle board, and MDF among others. Plywood, especially, is a ubiquitous construction material that varies in type and thicknesses and is readily available throughout the world. Its well documented history of use, ease of implementation and relatively low cost, all add to the feasibility of use. Automation has made mass production in a vast range of materials possible; wood is one of the simpler materials to process, hence there is a greater potential for the emergence of factories that can produce the RDA if it was constructed from plywood, ultimately making a system made from plywood more producible on a large scale. Furthermore, plywood could be manipulated in the field manually or with hand-held power tools. The aforementioned characteristics of plywood make it one of the more feasible materials to use in the field. As such, the forethought of plywood being one of the more feasible materials to implement informed the dimensional ratio, derived from its 48 inch width and 96 inch length, which was used for each material investigation.

Wood, cut along grain, has a modulus of elasticity of of approximately ($\lambda = \pm 1{,}600{,}000$ psi), and would fracture well before conforming to an acute angle. Accordingly, the system would have to be assembled from components as opposed to folding. Hinges or rigid fasteners could be used to connect the components depending on the desired performance of the system. The joint is inherently the weakest part of any connected multi-part system, as such, careful consideration must be paid to joint

detail. The wood to wood connection at the joint can however be substituted for metal to augment the strength of the joint. Eighth (⅛) inch steel straps can be pre-bent or manipulated in field, with relative ease, to make a rigid connection between parts.

Figure 21: Rigid Plate with Rigid Steel Connections

Figure 22: Bent ⅛" thick steel plate connection

CHAPTER 5: CONSTRUCTING THE VIRTUAL FOLD

Modeling the fold with virtual surfaces

Synchronized movement of the connected components is a direct result of folding as the configuration self-equalizes under force. Recreating an instance of the constructed fishbone pattern is a simple process; however recreating the fishbone system in the virtual realm requires an understanding of how the parts work to create the whole. Consequently, developing a virtual reverse fold started by sequentially modifying a right-angled trapezium by decreasing the length of shorter of the 2 parallel sides until its endpoints coincide, changing the trapezium to a triangle. This shape is used to generate the other quadrilaterals of the joint; the diagonal edge is shared between two vertically aligned adjacent trapeziums. Once the two trapeziums are reflected about the edge, all 4 components of the single joint are made. Diagramming the pleat entails incrementally rotating each half of the pleat about the main crease to produce a mountain fold. According to the analog model, this rotation induces a secondary rotation of the top pieces about their diagonal edge. A sequence of "snapshots" (*Figure 23*)of the systems movement were taken for each 5 degree rotation about the ridge for each iteration of the altered trapezium.

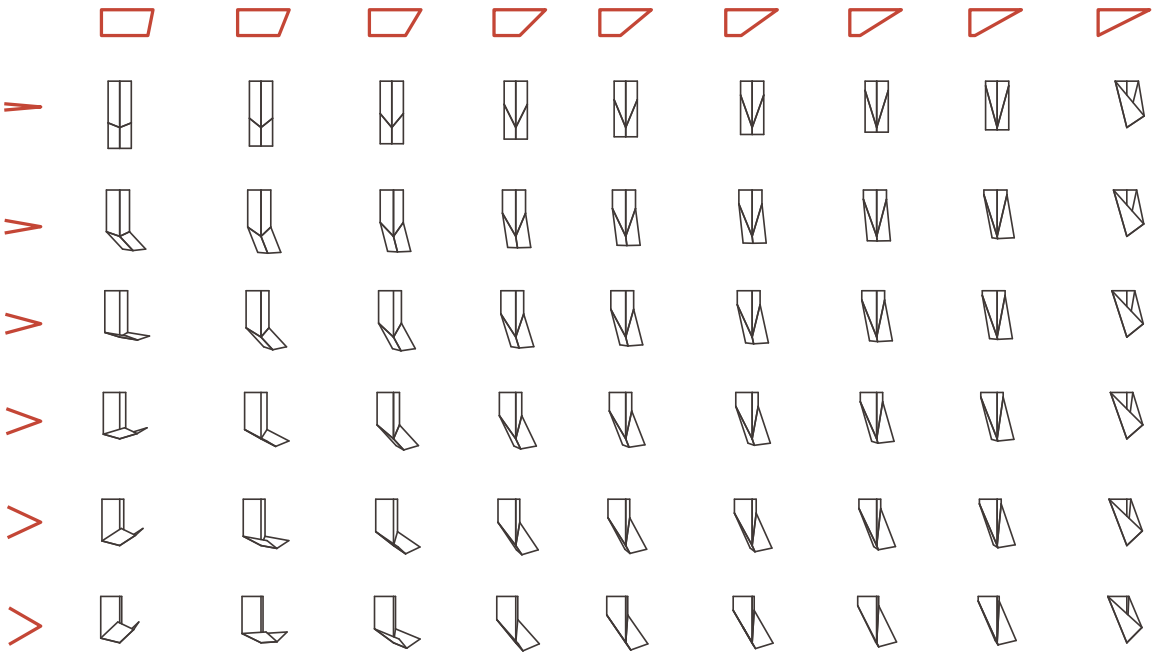

Figure 23: Permutations of a single reverse fold

Defining the Fold Parametrically

Based on the primary pleat algorithm the parameterized pleat system starts with the pleats which can vary in span based on the 3 points of the v-shaped cross section. Using Generative Components[9] to parameterize the pleat through scripting[10], requires establishing the relationship between the first and last points along a straight line that

9 A application founded on Bentley's MicroStation that facilitates the scripting of architecture.

10 In computer programming, a script is a program or sequence of instructions that is interpreted or carried out by another program rather than by the computer processor (as a compiled program is). What is a Script? - a definition from Whatis.com. <http://searchenterpriselinux.techtarget.com/sDefinition/0,,sid39_gci212948,00.html>

are used to generate the apex of the ridge which would fall in the middle of the starting points (*Figure 24*). The varying elevation of the apex point, maintains a predetermined distance to the defining end points of the cross section. Alternating the direction of the cross section along the profile of the structure determines the direction of each reverse fold along the building profile. Similarly, to the aforementioned implementation of the static reverse fold, the parameterized system is being informed by a single trapezium, which generates the other components of the system. Thus, only one side of the main crease needs to be explicitly defined.

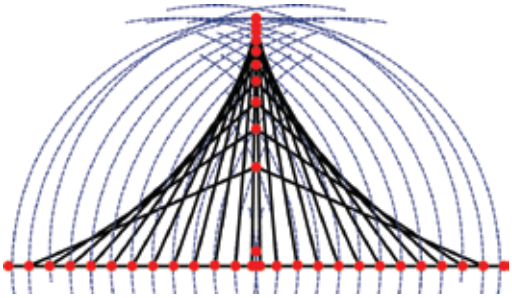

Figure 24: Constructing a Parametric Pleat Cross Section

The direction and length of the ridge line is defined by another point which is aligned with the apex of the pleat. The final vertex of the trapezium is defined by the intersection of a line projected from one of the cross section's end points and another line projected from the analogous point on the adjacent line segment. As a consequence of each trapezium is partially defined by another, allowing the adjustment of one to induce change in the adjacent parts. Because the span of the pleat remains constant and each shape helps to define the other through an intersection of lines parallel to its relative ridge line, the entire system can be automatically generated by recursively executing the steps for each line segment of the polygon (*Figure 25*).

Reflecting the trapeziums about the main crease generates the other half of the pleat which could then be duplicated to define a folded sheet. Manipulating the virtual surface is accomplished by moving the points which define the main crease of the pleat. The forms are the representations of the rule system of the fishbone pattern and as such, they are automatically updated. Therefore, multiple compartments can be composed through the arrangement of multiple profiles to induce furcations or monoliths.

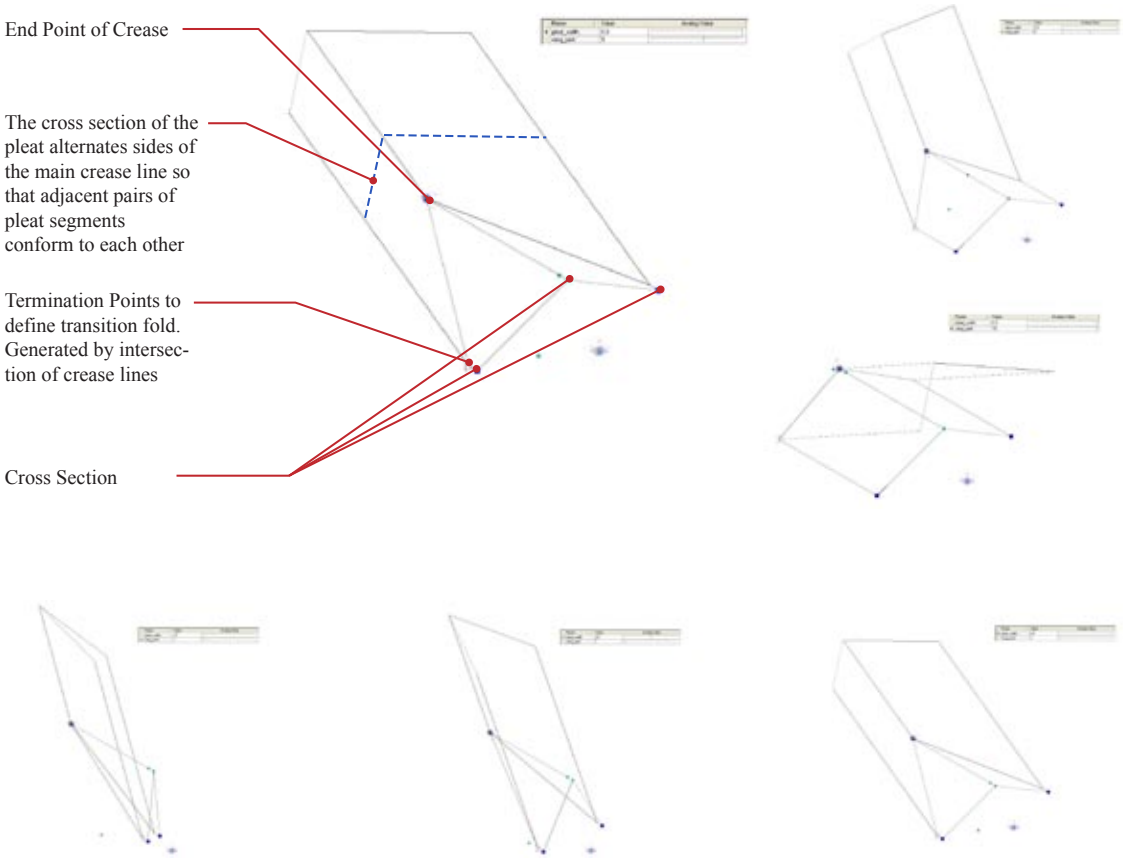

Figure 25: Establishing the reverse fold relationship with GC scripting

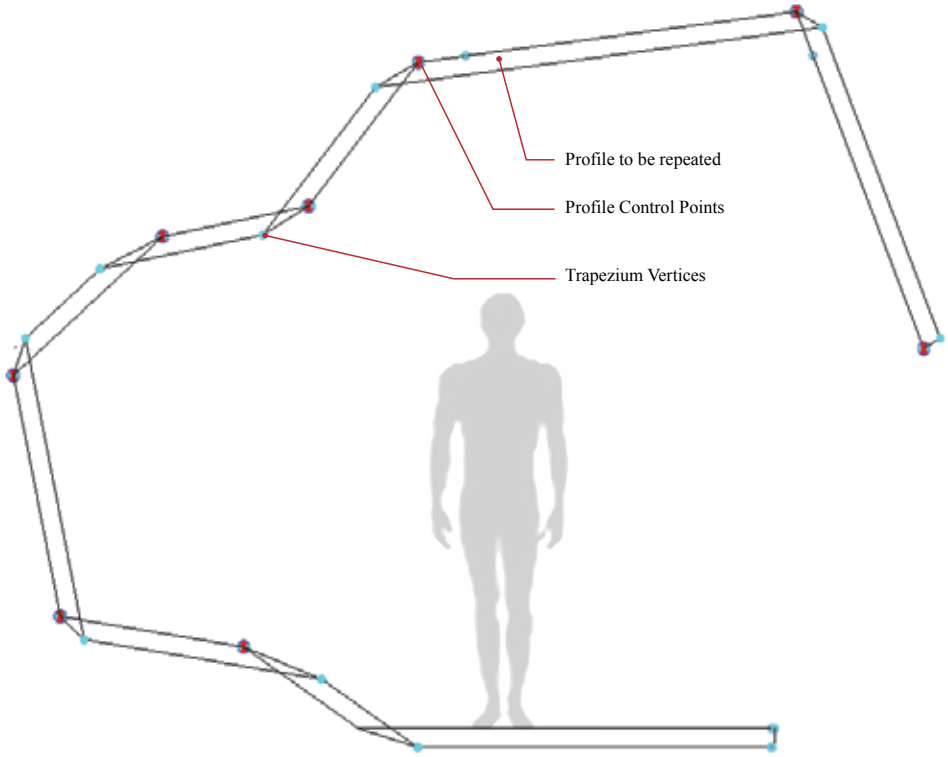

Figure 26: Pleat System Profile

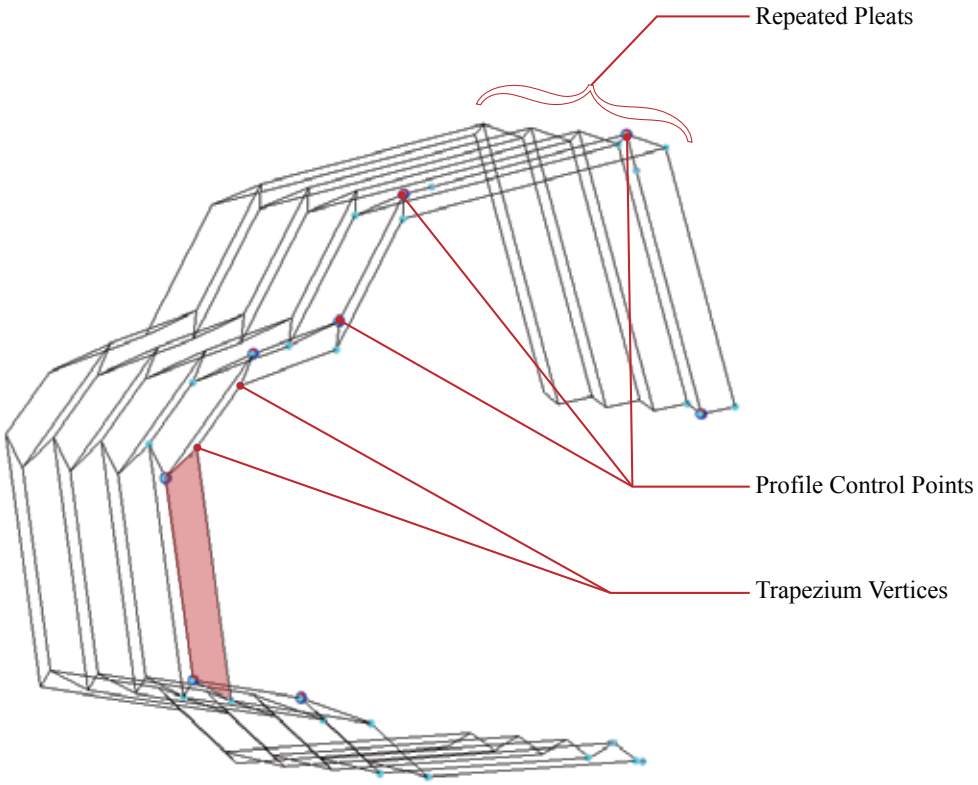

Figure 27: Pleat System Module

CHAPTER 6: FOLDING AS A DESIGN TECHNIQUE

Space and Structure

Inherited from the profile of the fishbone pleat manipulation, the profile of the RDA structure is defined by coplanar polygons that is derived from the two main creases of the pleat. The number of sides, their lengths and the angle between them are influenced by the function of the skin. Thus, the sectional definition of the spaces becomes as essential to the design as the floor plan. If the surface generated by the series of crease profiles is, in essence, an extrusion in the direction perpendicular to the plane of the shape (*Figure 28*), the RDA structure can be considered to be a loft using a series of curves that define a compound volume (*Figure 29*). It is a premeditated arrangement of extruded modules. It is this premeditation that allows the designer to introduce auxiliary functions, such as overhangs, niches, or furniture, to the design of the building envelope. The infusion of the folding and 'sheet materiality' into architecture facilitates the introduction of compound procedures. For example, paper's negligible thickness allows the designer to simultaneously manipulate multiple sheets of paper and then individually to generate space defining furcations. Architecturally, this can be used to generate buffers, corridors or larger spaces between spaces. The folding operations affect the surface functionally, aesthetically or as a combination of the two.

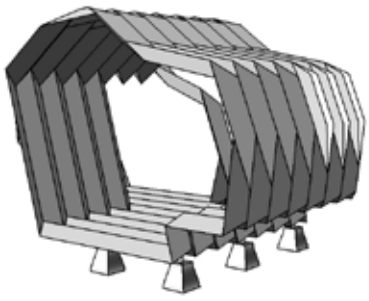

Figure 28: Example of an RDA Module

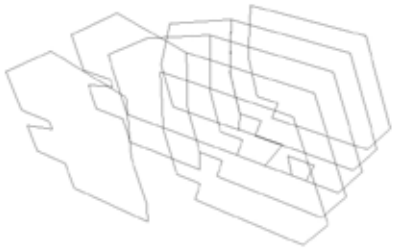

Figure 29: Series of coplanar polygons that define the RDA structure

Aesthetically, folding inherently addresses the notions of continuity and discontinuity of both surfaces and space. The two-sidedness of a planar material mean that surficial manipulations result in a binary translation between positive and negative

space. As a result, one face is the inverse of the other. This occurrence becomes more pronounced in a linearly informed surface: the niche and the outcropping are equal. A schism in the span of the surface occurs at the introduction of a new profile. The translations of the vertices on the new polygonal profile will inform the scale of the surficial change as and ultimately the volumetric configuration. A large enough change can produce an opening to a new room, or a subtle introduction of a cross sectional aperture. Reducing the wavelength of pleats increases the appearance of massiveness, while increasing the wavelength generates a sense of continuity.

By reducing the wavelength (*Figure 30*) of the pleats, the number of vertical pleats in a wall or similar vertical support condition can efficiently increase the load capacity of the support by increasing the density of material. Conversely, increasing the wavelength of pleats in a horizontal or spanning condition lightens the weight of material that has to be suspended while maintaining enough rigidity to span large distances.

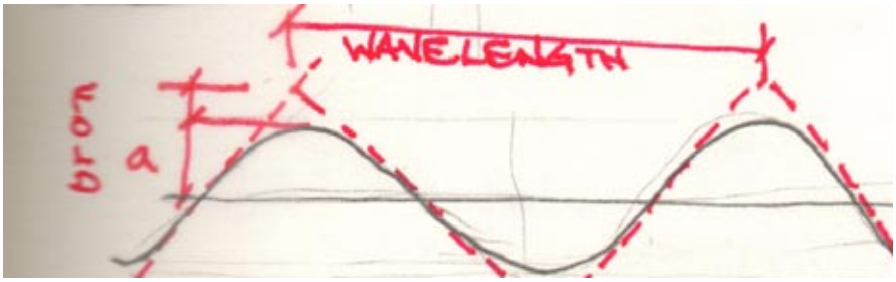

Figure 30: Wavelength of pleats

Texture and Aperture

Localizing movement to specific area is achieved by introducing disconnecting adjacent folds. Incisions along creases allow movement of the detached faces. The number of contiguous faces that is disconnected from the sheet informs the scale and performance of the aperture. As such, leaving two adjacent faces connected to each other by a main crease line but connected to the structure by opposing vertices will result in a folding louver type window. Whereas, leaving several faces in tact but connected to the whole by a crease creates a larger opening in the surface that is afforded by an isolated peeling away of faces.

Owing to the fact that the fishbone pleat is purely folded, the faces could be made from any planar material. Thus, any planar material could be cut to be substituted

for a face on the structure. Substitute various materials into the faces of the structure accommodates a discontinuity of visibility as well as textures. The size of the disruption is based on the contiguity of its respective material aggregations across the surface of the building; the contiguity of the material substitution is in turn driven by the performance of the enclosure or aestethic of the skin (*Figure 31*).

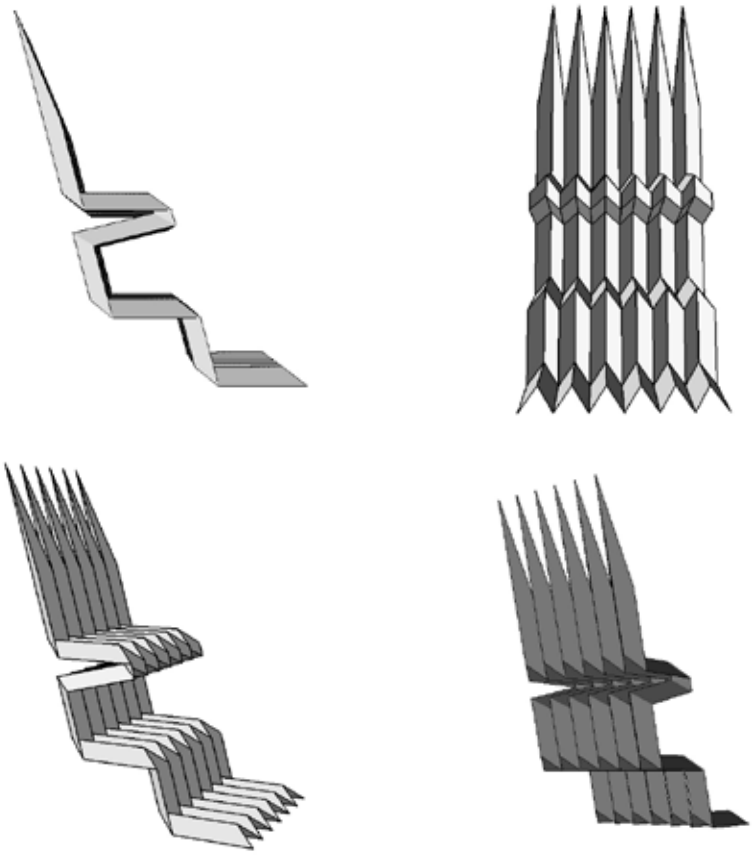

Figure 31: Example of profile used to inform stacked twin beds illustrating the surface's binary relationship between sides.

CHAPTER 7: CONCLUSION

By both the implicit and explicit definition of constraints to a problem, the designer can utilize a complex, and in some cases self referential, algorithm to generate an efficient solution based on a natural, self-equalizing phenomenon. The increase in computer's processing power has granted the capability to compute and produce feedback based results to the designer. This boost allows a highly recursive computation of the aforementioned complex morphogenetic computations. Not only does the scale less virtual world allow the designer to cross scales without a losing the precision of the calculation, but it also allows him to adjust the level of precision to address scale and resolution. Scripts, as produced by GC, are the designers interface with the solution. The designer can input the parameters of the problem, the design parameters, which the script guides through a mathematical algorithm to produce a solution that is based on a morphogenetic phenomenon. The parameters are used to define the controls of the generated artifact.

In this instance, folding operations were parameterized to define new rule set for the design of a transitional shelter. The rule set defines the words and the syntax that become the semantic components and syntactic assemblage of the architecture. As such, only a specific form and aesthetic can be accomplished by a specific morphogenetic routine. The parameterization of the fishbone pleat system generates an algorithm that the designer can use to develop a customizable, modular system of deployable architecture. The devised script produces an aggregation of trapeziums that are derived from the recursive generation reverse fold origami technique along a profile.

Building on the infrastructure produced by the previous script, it is possible for the designer to nest scripts to effect change in aesthetics or to introduce an additional performance attribute. Each overlaid layer of an algorithm increases the vocabulary of the rule set which, ultimately, facilitates a growth in the inherent architectural discourse of the system. Over time, natural evolution will introduce new words and syntax to the rule set to influence the discourse. As much as the innate layers can be being culturally influential, the newly introduced layers can be introduced by the culture of the inhabitants. Materials and methods of fabrication can induce changes that are cultural specific that remains specific to that site, time period or geographic location. Deploying the system in another crisis can yield different results that are based on the new parameters of design that are defined by the new context.

Continued advances in technology allow the designer to capitalize on its power

to develop a new language of design and construction through the re-appropriation of techniques, phenomena and software applications to facilitate the emergence of a new discourse.

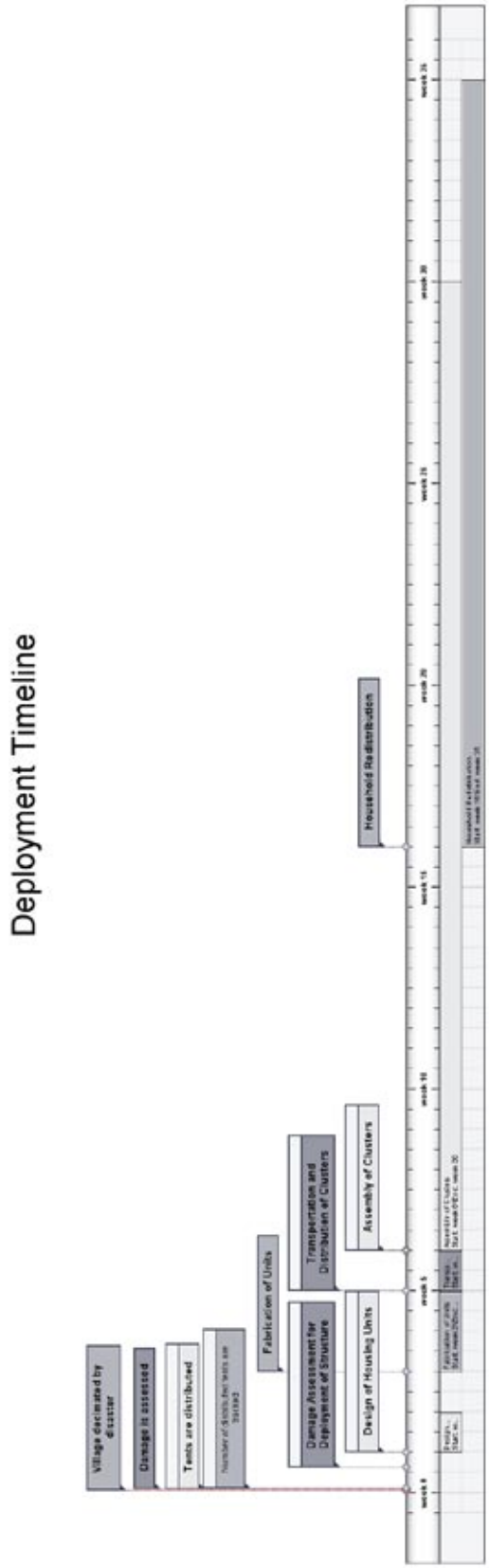

Figure 32: Deployment Timeline of the RDA System

APPENDIX A

GC Script: Fishbone Pleat Generator

```
transaction modelBased “Graph changed by user”
{
   feature bed_width GC.GraphVariable
   {
      Value                = 54.0;
      LimitValueToRange    = true;
      RangeMinimum         = 36.0;
      RangeMaximum         = 54.0;
      RangeStepSize        = 1.0;
   }
   feature line01 GC.Line
   {
      StartPoint           = point01;
      Direction            = baseCS.XDirection;
      Length               = 54;
   }
   feature point01 GC.Point
   {
      CoordinateSystem     = baseCS;
      XTranslation         = <free> (2.56473804700153);
      YTranslation         = <free> (-0.79194145750439);
      ZTranslation         = <free> (0.0);
      HandlesVisible       = true;
   }
}

transaction modelBased “Graph changed by user”
{
   feature arc01 GC.Arc
   {
      CenterPoint          = point03;
      Support              = baseCS.XZPlane;
      Radius               = 10;
      SweepAngle           = 180;
   }
   feature line01 GC.Line
   {
      Length               = bed_width;
   }
   feature point01 GC.Point
   {
```

```
    Replication          = ReplicationOption.AllCombinations;
    Visible              = false;
  }
  feature point03 GC.Point
  {
    Curve                = line01;
    T                    = Series(0.0, 1.0, 0.125);
    HandlesVisible       = true;
    Replication          = ReplicationOption.CorrespondingIndexing;
  }
}

transaction modelBased "Graph changed by user"
{
  feature arc01 GC.Arc
  {
    StartAngle           = -90;
    Visible              = false;
  }
  feature point02 GC.Point
  {
    Curve0               = arc01[1];
    Curve1               = arc01[0];
  }
  feature point04 GC.Point
  {
    Curve0               = arc01[1];
    Curve1               = arc01[2];
  }
  feature point05 GC.Point
  {
    Curve0               = arc01[3];
    Curve1               = arc01[2];
  }
  feature point06 GC.Point
  {
    Curve0               = arc01[3];
    Curve1               = arc01[4];
  }
  feature point07 GC.Point
  {
    Curve0               = arc01[5];
    Curve1               = arc01[4];
  }
  feature point08 GC.Point
  {
```

```
      Curve0              = arc01[5];
      Curve1              = arc01[6];
   }
   feature point09 GC.Point
   {
      Curve0              = arc01[7];
      Curve1              = arc01[6];
   }
   feature point10 GC.Point
   {
      Curve0              = arc01[7];
      Curve1              = arc01[8];
   }
}

transaction modelBased “Graph changed by user”
{
   feature line01 GC.Line
   {
      Visible             = false;
   }
   feature polyLine01 GC.PolyLine
   {
      Vertices            = {point03[0],point02,point03[1],point04,point03[2],point05,
point03[3],point06,point03[4],point07,point03[5],point08,point03[6],point09,point03[
7],point10,point03[8]};
   }
}

transaction modelBased “Graph changed by user”
{
   feature arc01 GC.Arc
   {
      SymbolXY            = {98, 108};
   }
   feature line01 GC.Line
   {
      SymbolXY            = {102, 103};
   }
   feature point03 GC.Point
   {
      SymbolXY            = {99, 105};
   }
   feature polyLine01 GC.PolyLine
   {
      SymbolXY            = {104, 104};
```

```
    }
}

transaction modelBased “Graph changed by user”
{
    feature point03 GC.Point
    {
        T                   = Series(0.0, 1.0, 0.25);
    }
    feature polyLine01 GC.PolyLine
    {
        Vertices            = {point03[0],point02,point03[1],point04,point03[2],point05,poi
nt03[3],point06,point03[4]};
    }
}

transaction modelBased “Graph changed by user”
{
    feature coordinateSystem01 GC.CoordinateSystem
    {
        Curve               = line02;
        T                   = 1;
        HandlesVisible      = true;
    }
    feature floor GC.GraphVariable
    {
        Value               = 96;
        LimitValueToRange   = true;
        RangeMinimum        = 48.0;
        RangeMaximum        = 96.0;
        RangeStepSize       = 6.0;
    }
    feature line02 GC.Line
    {
        StartPoint          = point02;
        Direction           = baseCS.YDirection;
        Length              = floor;
    }
    feature point11 GC.Point
    {
        CoordinateSystem    = coordinateSystem01;
        XTranslation        = 0;
        YTranslation        = 0;
        ZTranslation        = 12;
    }
}
```

```
transaction modelBased “Graph changed by user”
{
   feature coordinateSystem02 GC.CoordinateSystem
   {
      Origin                = point12;
      CoordinateSystem      = coordinateSystem01;
      RotationAngle         = 135;
      Axis                  = AxisOption.Z;
   }
   feature line03 GC.Line
   {
      StartPoint            = point11;
      EndPoint              = line02.EndPoint;
   }
   feature line04 GC.Line
   {
      StartPoint            = point03[0];
      Direction             = baseCS.YDirection;
      Length                = 4;
   }
   feature point12 GC.Point
   {
      Curve                 = line03;
      T                     = <free> (0.705624024436407);
      HandlesVisible        = true;
   }
}

transaction modelBased “Graph changed by user”
{
   feature coordinateSystem03 GC.CoordinateSystem
   {
      Origin                = point02;
      PrimaryDirection      = point13;
      PrimaryAxis           = AxisOption.Z;
      SecondaryDirection    = point03[1];
      SecondaryAxis         = AxisOption.X;
   }
   feature point13 GC.Point
   {
      Curve                 = line02;
      T                     = <free> (0.988489876961843);
      HandlesVisible        = true;
   }
}
```

```
transaction modelBased “Graph changed by user”
{
   feature coordinateSystem02 GC.CoordinateSystem
   {
      RotationAngle            = -45;
   }
}

transaction modelBased “Graph changed by user”
{
   deleteFeature coordinateSystem03;
   deleteFeature point13;
   feature coordinateSystem02 GC.CoordinateSystem
   {
      RotationAngle            = -90;
      Axis                     = AxisOption.Y;
   }
}

transaction modelBased “Graph changed by user”
{
   feature point14 GC.Point
   {
      Plane                    = baseCS.XZPlane;
      PointToProjectOnToPlane  = point03[0];
   }
   feature point15 GC.Point
   {
      Plane                    = baseCS.XZPlane;
      PointToProjectOnToPlane  = point02;
   }
   feature point16 GC.Point
   {
      Plane                    = baseCS.XZPlane;
      PointToProjectOnToPlane  = point03[1];
   }
}

transaction modelBased “Graph changed by user”
{
   deleteFeature coordinateSystem02;
   deleteFeature point12;
   feature coordinateSystem04 GC.CoordinateSystem
   {
      Curve                    = line03;
```

```
    T                    = <free> (0.842627654410273);
    HandlesVisible       = true;
  }
  feature coordinateSystem05 GC.CoordinateSystem
  {
    Origin               = point15;
    PrimaryDirection     = point14;
    PrimaryAxis          = AxisOption.X;
    SecondaryDirection   = point16;
    SecondaryAxis        = AxisOption.Y;
  }
  feature coordinateSystem06 GC.CoordinateSystem
  {
    CoordinateSystem     = coordinateSystem04;
    XTranslation         = -4;
    YTranslation         = 0;
    ZTranslation         = 0;
    XRotation            = 45;
    YRotation            = 90;
    ZRotation            = 90;
  }
  feature point11 GC.Point
  {
    XTranslation         = <free> (0.0);
    ZTranslation         = <free> (12);
  }
}

transaction modelBased "Graph changed by user"
{
  feature coordinateSystem06 GC.CoordinateSystem
  {
    XRotation            = -135;
    ZRotation            = -270;
  }
  feature line05 GC.Line
  {
    StartPoint           = point18;
    Direction            = coordinateSystem06.ZDirection;
    Length               = 1;
  }
  feature line06 GC.Line
  {
    StartPoint           = point19;
    Direction            = coordinateSystem06.ZDirection;
    Length               = 1;
```

```
    }
    feature line07 GC.Line
    {
      StartPoint              = point16;
      Direction               = baseCS.YDirection;
      Length                  = 4;
    }
    feature point17 GC.Point
    {
      FeatureToCopy           = point15;
      CopyFrom                = coordinateSystem05;
      CopyTo                  = coordinateSystem06;
    }
    feature point18 GC.Point
    {
      FeatureToCopy           = point14;
      CopyFrom                = coordinateSystem05;
      CopyTo                  = coordinateSystem06;
    }
    feature point19 GC.Point
    {
      FeatureToCopy           = point16;
      CopyFrom                = coordinateSystem05;
      CopyTo                  = coordinateSystem06;
    }
    feature point20 GC.Point
    {
      Curve0                  = line06;
      Curve1                  = line04;
    }
    feature point21 GC.Point
    {
      Curve0                  = line05;
      Curve1                  = line07;
    }
    feature polygon01 GC.Polygon
    {
      Vertices                = {point03[0],point02,line02.EndPoint,point20};
    }
}

transaction modelBased “Graph changed by user”
{
    feature polygon02 GC.Polygon
    {
      Vertices                = {point03[1],point02,line02.EndPoint,point21};
```

```
    }
}

transaction modelBased “Graph changed by user”
{
    {
        Value                   = 36.0;
    }
    feature coordinateSystem07 GC.CoordinateSystem
    {
        Origin                  = point15;
        PrimaryDirection        = baseCS.YDirection;
        PrimaryAxis             = AxisOption.Z;
        SecondaryDirection      = baseCS.XDirection;
        SecondaryAxis           = AxisOption.X;
    }
    feature direction01 GC.Direction
    {
        Origin                  = coordinateSystem01;
        DirectionPoint          = point11;
    }
    feature direction03 GC.Direction
    {
        Origin                  = point28;
        DirectionPoint          = point11;
    }
    feature line10 GC.Line
    {
        StartPoint              = point28;
        EndPoint                = point11;
    }
    feature point11 GC.Point
    {
        ZTranslation            = <free> (12.0);
    }
    feature point28 GC.Point
    {
        CoordinateSystem        = coordinateSystem01;
        XTranslation            = <free> (0.0);
        YTranslation            = 0;
        ZTranslation            = 30;
    }
}

transaction modelBased “Graph changed by user”
{
```

```
feature coordinateSystem01 GC.CoordinateSystem
{
   Visible                = false;
}
feature coordinateSystem08 GC.CoordinateSystem
{
   Origin                 = point22;
   PrimaryDirection         = direction01;
   PrimaryAxis              = AxisOption.Z;
   SecondaryDirection       = baseCS.XDirection;
   SecondaryAxis            = AxisOption.X;
   Visible                = false;
}
feature coordinateSystem09 GC.CoordinateSystem
{
   CoordinateSystem         = coordinateSystem08;
   XTranslation             = 0;
   YTranslation             = 0;
   ZTranslation             = 2;
   XRotation                = 0;
   YRotation                = 0;
   ZRotation                = 180;
}
feature direction01 GC.Direction
{
   Visible                = false;
}
feature line08 GC.Line
{
   StartPoint               = point24;
   Direction                = coordinateSystem09.ZDirection;
   Length                   = -1;
}
feature line09 GC.Line
{
   StartPoint               = point23;
   Direction                = coordinateSystem09.ZDirection;
   Length                   = -1;
}
feature point22 GC.Point
{
   Curve                    = line03;
   T                      = <free> (0.777485391423938);
   HandlesVisible           = true;
   Visible                = false;
}
```

```
    feature point23 GC.Point
    {
      FeatureToCopy          = point14;
      CopyFrom               = coordinateSystem07;
      CopyTo                 = coordinateSystem09;
    }
    feature point24 GC.Point
    {
      FeatureToCopy          = point16;
      CopyFrom               = coordinateSystem07;
      CopyTo                 = coordinateSystem09;
    }
    feature point25 GC.Point
    {
      Curve0                 = line04;
      Curve1                 = line08;
    }
    feature point26 GC.Point
    {
      FeatureToCopy          = point15;
      CopyFrom               = coordinateSystem07;
      CopyTo                 = coordinateSystem09;
    }
    feature point27 GC.Point
    {
      Curve0                 = line09;
      Curve1                 = line07;
    }
    feature polygon03 GC.Polygon
    {
      Vertices               = {point03[0],point02,line02.EndPoint,point25};
    }
    feature polygon04 GC.Polygon
    {
      Vertices               = {point03[1],point02,line02.EndPoint,point27};
    }
}

transaction modelBased “Graph changed by user”
{
    feature coordinateSystem09 GC.CoordinateSystem
    {
      Visible                = false;
    }
    feature coordinateSystem10 GC.CoordinateSystem
    {
```

```
    Origin                  = point29;
    PrimaryDirection        = direction03;
    PrimaryAxis             = AxisOption.Z;
    SecondaryDirection      = baseCS.XDirection;
    SecondaryAxis           = AxisOption.X;
    Visible                 = false;
}
feature coordinateSystem11 GC.CoordinateSystem
{
    CoordinateSystem        = coordinateSystem10;
    XTranslation            = 0;
    YTranslation            = 0;
    ZTranslation            = -2;
    XRotation               = 0;
    YRotation               = 0;
    ZRotation               = 180;
}
feature line08 GC.Line
{
    SymbolXY                = {101, 134};
}
feature line11 GC.Line
{
    StartPoint              = point31;
    Direction               = coordinateSystem11.ZDirection;
    Length                  = 1;
}
feature line12 GC.Line
{
    StartPoint              = point30;
    Direction               = coordinateSystem11.ZDirection;
    Length                  = 1;
}
feature point11 GC.Point
{
    XTranslation            = <free> (7.81135418738108);
}
feature point28 GC.Point
{
    XTranslation            = <free> (-18.1848022736536);
}
feature point29 GC.Point
{
    Curve                   = line10;
    T                       = <free> (0.831637012775693);
    HandlesVisible          = true;
```

```
    Visible                = false;
  }
  feature point30 GC.Point
  {
    FeatureToCopy          = point23;
    CopyFrom               = coordinateSystem09;
    CopyTo                 = coordinateSystem11;
  }
  feature point31 GC.Point
  {
    FeatureToCopy          = point24;
    CopyFrom               = coordinateSystem09;
    CopyTo                 = coordinateSystem11;
  }
  feature point32 GC.Point
  {
    FeatureToCopy          = point26;
    CopyFrom               = coordinateSystem09;
    CopyTo                 = coordinateSystem11;
  }
  feature point33 GC.Point
  {
    Curve0                 = line11;
    Curve1                 = line08;
  }
  feature point34 GC.Point
  {
    Curve0                 = line12;
    Curve1                 = line09;
  }
  feature polygon05 GC.Polygon
  {
    Vertices               = {point25,line02.EndPoint,point11,point33};
  }
  feature polygon06 GC.Polygon
  {
    Vertices               = {point27,line02.EndPoint,point11,point34};
  }
}

transaction modelBased “Graph changed by user”
{
  feature coordinateSystem12 GC.CoordinateSystem
  {
    Origin                 = point36;
    PrimaryDirection       = direction02;
```

```
   PrimaryAxis              = AxisOption.Z;
   SecondaryDirection       = baseCS.XDirection;
   SecondaryAxis            = AxisOption.X;
   Visible                  = false;
}
feature coordinateSystem13 GC.CoordinateSystem
{
   CoordinateSystem         = coordinateSystem12;
   XTranslation             = 0;
   YTranslation             = 0;
   ZTranslation             = -2;
   XRotation                = 0;
   YRotation                = 0;
   ZRotation                = 180;
}
feature direction02 GC.Direction
{
   Origin                   = point28;
   DirectionPoint           = point35;
}
feature line13 GC.Line
{
   StartPoint               = point35;
   EndPoint                 = point28;
}
feature line14 GC.Line
{
   StartPoint               = point37;
   Direction                = coordinateSystem13.ZDirection;
   Length                   = 1;
}
feature line15 GC.Line
{
   StartPoint               = point38;
   Direction                = coordinateSystem13.ZDirection;
   Length                   = 1;
}
feature point11 GC.Point
{
   ZTranslation             = <free> (35.9885314374329);
}
feature point28 GC.Point
{
   XTranslation             = <free> (-17.8753724757053);
   ZTranslation             = <free> (74.4040922153327);
}
```

```
    feature point35 GC.Point
    {
      CoordinateSystem        = coordinateSystem01;
      XTranslation            = <free> (7.81135418738108);
      YTranslation            = 0;
      ZTranslation            = <free> (91.6578268063848);
    }
    feature point36 GC.Point
    {
      Curve                   = line13;
      T                       = <free> (0.822280243514969);
      HandlesVisible          = true;
      Visible                 = false;
    }
    feature point37 GC.Point
    {
      FeatureToCopy           = point31;
      CopyFrom                = coordinateSystem11;
      CopyTo                  = coordinateSystem13;
    }
    feature point38 GC.Point
    {
      FeatureToCopy           = point30;
      CopyFrom                = coordinateSystem11;
      CopyTo                  = coordinateSystem13;
    }
    feature point39 GC.Point
    {
      Curve0                  = line11;
      Curve1                  = line14;
    }
    feature point40 GC.Point
    {
      Curve0                  = line12;
      Curve1                  = line15;
    }
    feature polygon07 GC.Polygon
    {
      Vertices                = {point11,point33,point39,point28};
    }
    feature polygon08 GC.Polygon
    {
      Vertices                = {point11,point34,point40,point28};
    }
}
```

```
transaction modelBased “Graph changed by user”
{
   feature bed_width GC.GraphVariable
   {
      Value                = 72.0;
      RangeMinimum         = 24.0;
      RangeMaximum         = 72.0;
      RangeStepSize        = 2.0;
   }
   feature line14 GC.Line
   {
      SymbolXY             = {101, 148};
   }
   feature point28 GC.Point
   {
      ZTranslation         = <free> (32.3960095828919);
   }
}

transaction modelBased “Graph changed by user”
{
   feature coordinateSystem14 GC.CoordinateSystem
   {
      Origin               = point42;
      PrimaryDirection     = direction04;
      PrimaryAxis          = AxisOption.Z;
      SecondaryDirection   = baseCS.XDirection;
      SecondaryAxis        = AxisOption.X;
      Visible              = false;
   }
   feature coordinateSystem15 GC.CoordinateSystem
   {
      CoordinateSystem     = coordinateSystem14;
      XTranslation         = 0;
      YTranslation         = 0;
      ZTranslation         = -3;
      XRotation            = 0;
      YRotation            = 0;
      ZRotation            = 180;
   }
   feature direction04 GC.Direction
   {
      Origin               = point41;
      DirectionPoint       = point35;
   }
   feature line16 GC.Line
```

```
{
   StartPoint              = point41;
   EndPoint                = point35;
}
feature line17 GC.Line
{
   StartPoint              = point43;
   Direction               = coordinateSystem15.ZDirection;
   Length                  = 1;
}
feature line18 GC.Line
{
   StartPoint              = point44;
   Direction               = coordinateSystem15.ZDirection;
   Length                  = 1;
}
feature point28 GC.Point
{
   ZTranslation            = <free> (46.4477462980562);
}
feature point41 GC.Point
{
   CoordinateSystem        = coordinateSystem01;
   XTranslation            = <free> (7.81135418738108);
   YTranslation            = 0;
   ZTranslation            = <free> (144);
}
feature point42 GC.Point
{
   Curve                   = line16;
   T                       = <free> (0.955467038209831);
   HandlesVisible          = true;
   Visible                 = false;
}
feature point43 GC.Point
{
   FeatureToCopy           = point37;
   CopyFrom                = coordinateSystem13;
   CopyTo                  = coordinateSystem15;
}
feature point44 GC.Point
{
   FeatureToCopy           = point38;
   CopyFrom                = coordinateSystem13;
   CopyTo                  = coordinateSystem15;
}
```

```
    feature point45 GC.Point
    {
        Curve0              = line14;
        Curve1              = line17;
    }
    feature point46 GC.Point
    {
        Curve0              = line18;
        Curve1              = line15;
    }
    feature polygon09 GC.Polygon
    {
        Vertices            = {point28,point39,point45,point35};
    }
    feature polygon10 GC.Polygon
    {
        Vertices            = {point28,point40,point46,point35};
    }
}

transaction modelBased “Graph changed by user”
{
    feature point28 GC.Point
    {
        XTranslation        = <free> (22.1811092179064);
    }
}

transaction modelBased “Graph changed by user”
{
    feature coordinateSystem11 GC.CoordinateSystem
    {
        Visible             = false;
    }
    feature coordinateSystem13 GC.CoordinateSystem
    {
        Visible             = false;
    }
    feature coordinateSystem15 GC.CoordinateSystem
    {
        Visible             = false;
    }
    feature coordinateSystem16 GC.CoordinateSystem
    {
        Origin              = point48;
        PrimaryDirection    = direction05;
```

```
    PrimaryAxis             = AxisOption.Z;
    SecondaryDirection      = baseCS.XDirection;
    SecondaryAxis           = AxisOption.X;
    Visible                 = false;
}
feature coordinateSystem17 GC.CoordinateSystem
{
    CoordinateSystem        = coordinateSystem16;
    XTranslation            = 0;
    YTranslation            = 0;
    ZTranslation            = 3;
    XRotation               = 0;
    YRotation               = 0;
    ZRotation               = 180;
    Visible                 = false;
}
feature coordinateSystem18 GC.CoordinateSystem
{
    Origin                  = point54;
    PrimaryDirection        = direction06;
    PrimaryAxis             = AxisOption.Z;
    SecondaryDirection      = baseCS.XDirection;
    SecondaryAxis           = AxisOption.X;
    Visible                 = false;
}
feature coordinateSystem19 GC.CoordinateSystem
{
    CoordinateSystem        = coordinateSystem18;
    XTranslation            = 0;
    YTranslation            = 0;
    ZTranslation            = -3;
    XRotation               = 0;
    YRotation               = 0;
    ZRotation               = 180;
    Visible                 = false;
}
feature coordinateSystem20 GC.CoordinateSystem
{
    Origin                  = point60;
    PrimaryDirection        = direction07;
    PrimaryAxis             = AxisOption.Z;
    SecondaryDirection      = baseCS.XDirection;
    SecondaryAxis           = AxisOption.X;
    Visible                 = false;
}
feature coordinateSystem21 GC.CoordinateSystem
```

```
{
   CoordinateSystem         = coordinateSystem20;
   XTranslation             = 0;
   YTranslation             = 0;
   ZTranslation             = -3;
   XRotation                = 0;
   YRotation                = 0;
   ZRotation                = 180;
}
feature direction05 GC.Direction
{
   Origin                   = point41;
   DirectionPoint           = point47;
}
feature direction06 GC.Direction
{
   Origin                   = point53;
   DirectionPoint           = point47;
}
feature direction07 GC.Direction
{
   Origin                   = point53;
   DirectionPoint           = point59;
}
feature floor GC.GraphVariable
{
   Value                    = 66.0;
}
feature line08 GC.Line
{
   Visible                  = false;
}
feature line09 GC.Line
{
   Visible                  = false;
}
feature line11 GC.Line
{
   Visible                  = false;
}
feature line12 GC.Line
{
   Visible                  = false;
}
feature line14 GC.Line
{
```

```
   Visible              = false;
}
feature line15 GC.Line
{
   Visible              = false;
}
feature line18 GC.Line
{
   Visible              = false;
}
feature line19 GC.Line
{
   StartPoint           = point47;
   EndPoint             = point41;
}
feature line20 GC.Line
{
   StartPoint           = point49;
   Direction            = coordinateSystem17.ZDirection;
   Length               = -1;
}
feature line21 GC.Line
{
   StartPoint           = point50;
   Direction            = coordinateSystem17.ZDirection;
   Length               = -1;
}
feature line22 GC.Line
{
   StartPoint           = point53;
   EndPoint             = point47;
}
feature line23 GC.Line
{
   StartPoint           = point55;
   Direction            = coordinateSystem19.ZDirection;
   Length               = 1;
   Visible              = false;
}
feature line24 GC.Line
{
   StartPoint           = point56;
   Direction            = coordinateSystem19.ZDirection;
   Length               = 1;
   Visible              = false;
}
```

```
feature line25 GC.Line
{
   StartPoint              = point53;
   EndPoint                = point59;
}
feature line26 GC.Line
{
   StartPoint              = point62;
   Direction               = coordinateSystem21.ZDirection;
   Length                  = -1;
}
feature line27 GC.Line
{
   StartPoint              = point61;
   Direction               = coordinateSystem21.ZDirection;
   Length                  = -1;
}
feature point23 GC.Point
{
   Visible                 = false;
}
feature point24 GC.Point
{
   Visible                 = false;
}
feature point26 GC.Point
{
   Visible                 = false;
}
feature point30 GC.Point
{
   Visible                 = false;
}
feature point31 GC.Point
{
   Visible                 = false;
}
feature point32 GC.Point
{
   Visible                 = false;
}
feature point35 GC.Point
{
   XTranslation            = <free> (-33.9981963132696);
}
feature point37 GC.Point
```

```
{
   Visible                = false;
}
feature point38 GC.Point
{
   Visible                = false;
}
feature point41 GC.Point
{
   XTranslation            = <free> (20.4338379280646);
   ZTranslation            = <free> (132.942206311559);
}
feature point43 GC.Point
{
   Visible                = false;
}
feature point44 GC.Point
{
   Visible                = false;
}
feature point47 GC.Point
{
   CoordinateSystem        = coordinateSystem01;
   XTranslation            = <free> (7.81135418738108);
   YTranslation            = 0;
   ZTranslation            = <free> (180);
}
feature point48 GC.Point
{
   Curve                   = line19;
   T                     = <free> (0.865764630783267);
   HandlesVisible          = true;
   Visible                = false;
}
feature point49 GC.Point
{
   FeatureToCopy           = point43;
   CopyFrom                = coordinateSystem15;
   CopyTo                  = coordinateSystem17;
   Visible                = false;
}
feature point50 GC.Point
{
   FeatureToCopy           = point44;
   CopyFrom                = coordinateSystem15;
   CopyTo                  = coordinateSystem17;
```

```
   Visible                = false;
}
feature point51 GC.Point
{
   Curve0                 = line20;
   Curve1                 = line17;
}
feature point52 GC.Point
{
   Curve0                 = line21;
   Curve1                 = line18;
}
feature point53 GC.Point
{
   CoordinateSystem         = coordinateSystem01;
   XTranslation             = <free> (24.3109325456868);
   YTranslation             = 0;
   ZTranslation             = <free> (225);
}
feature point54 GC.Point
{
   Curve                  = line22;
   T                    = <free> (0.899053838417365);
   HandlesVisible           = true;
   Visible                = false;
}
feature point55 GC.Point
{
   FeatureToCopy            = point49;
   CopyFrom                 = coordinateSystem17;
   CopyTo                 = coordinateSystem19;
   Visible                = false;
}
feature point56 GC.Point
{
   FeatureToCopy            = point50;
   CopyFrom                 = coordinateSystem17;
   CopyTo                 = coordinateSystem19;
   Visible                = false;
}
feature point57 GC.Point
{
   Curve0                 = line20;
   Curve1                 = line23;
}
feature point58 GC.Point
```

```
{
   Curve0                = line24;
   Curve1                = line21;
}
feature point59 GC.Point
{
   CoordinateSystem        = coordinateSystem01;
   XTranslation            = <free> (24.3109325456868);
   YTranslation            = 0;
   ZTranslation            = 260;
}
feature point60 GC.Point
{
   Curve                 = line25;
   T                     = <free> (0.159264346671556);
   HandlesVisible          = true;
   Visible               = false;
}
feature point61 GC.Point
{
   FeatureToCopy           = point56;
   CopyFrom              = coordinateSystem19;
   CopyTo                = coordinateSystem21;
}
feature point62 GC.Point
{
   FeatureToCopy           = point55;
   CopyFrom              = coordinateSystem19;
   CopyTo                = coordinateSystem21;
}
feature point63 GC.Point
{
   Curve0                = line27;
   Curve1                = line24;
}
feature point64 GC.Point
{
   Curve0                = line26;
   Curve1                = line23;
}
feature polygon05 GC.Polygon
{
   Fill                  = true;
}
feature polygon06 GC.Polygon
{
```

```
   Fill                = true;
}
feature polygon07 GC.Polygon
{
   Fill                = true;
}
feature polygon08 GC.Polygon
{
   Fill                = true;
}
feature polygon09 GC.Polygon
{
   Fill                = true;
}
feature polygon10 GC.Polygon
{
   Fill                = true;
}
feature polygon11 GC.Polygon
{
   Vertices              = {point35,point45,point51,point41};
   Fill                = true;
}
feature polygon12 GC.Polygon
{
   Vertices              = {point46,point35,point41,point52};
   Fill                = true;
}
feature polygon13 GC.Polygon
{
   Vertices              = {point51,point41,point47,point57};
   Fill                = true;
}
feature polygon14 GC.Polygon
{
   Vertices              = {point52,point41,point47,point58};
   Fill                = true;
}
feature polygon15 GC.Polygon
{
   Vertices              = {point57,point47,point53,point64};
   Fill                = true;
}
feature polygon16 GC.Polygon
{
   Vertices              = {point58,point63,point53,point47};
```

```
      Fill                  = true;
   }
}

transaction modelBased “Graph changed by user”
{
   feature point11 GC.Point
   {
      XTranslation          = <free> (39.6683074318729);
      ZTranslation          = <free> (10.5547268532239);
   }
   feature point28 GC.Point
   {
      XTranslation          = <free> (21.0702867005678);
      ZTranslation          = <free> (43.9652263153856);
   }
   feature point35 GC.Point
   {
      XTranslation          = <free> (-22.4075975492132);
      ZTranslation          = <free> (41.6597377811623);
   }
   feature point41 GC.Point
   {
      XTranslation          = <free> (48.8802787781626);
      ZTranslation          = <free> (66.2933274522871);
   }
   feature point47 GC.Point
   {
      XTranslation          = <free> (71.3810737410913);
      ZTranslation          = <free> (126.001244460882);
   }
   feature point53 GC.Point
   {
      ZTranslation          = <free> (164.878265878029);
   }
   feature point59 GC.Point
   {
      ZTranslation          = <free> (260);
   }
}

transaction modelBased “Graph changed by user”
{
   feature polygon17 GC.Polygon
   {
      FeatureToCopy         = polygon03;
```

```
      CopyFrom              = point02;
      CopyTo                = {point04,point05,point06};
   }
}

transaction modelBased “Graph changed by user”
{
   feature polygon18 GC.Polygon
   {
      FeatureToCopy         = polygon04;
      CopyFrom              = point02;
      CopyTo                = {point04,point05,point06};
   }
}

transaction modelBased “Graph changed by user”
{
   feature point11 GC.Point
   {
      XTranslation          = <free> (15.4124848043914);
      ZTranslation          = <free> (7.94355981844541);
   }
   feature point35 GC.Point
   {
      XTranslation          = <free> (4.75370602287643);
      ZTranslation          = <free> (55.4129773632376);
   }
   feature point47 GC.Point
   {
      XTranslation          = <free> (33.5758385600455);
      ZTranslation          = <free> (147.205875098659);
   }
   feature point53 GC.Point
   {
      XTranslation          = <free> (-82.5627375487281);
      ZTranslation          = <free> (132.379113743597);
   }
   feature polygon19 GC.Polygon
   {
      FeatureToCopy         = {polygon05,polygon06,polygon07,polygon08,polygon09,p
olygon10,polygon11,polygon12,polygon13,polygon14,polygon15,polygon16};
      CopyFrom              = point02;
      CopyTo                = point04;
   }
   feature polygon20 GC.Polygon
   {
```

```
      FeatureToCopy          = {polygon05,polygon06,polygon07,polygon08,polygon09,p
olygon10,polygon11,polygon12,polygon13,polygon14,polygon15,polygon16};
      CopyFrom               = point02;
      CopyTo                 = point05;
   }
   feature polygon21 GC.Polygon
   {
      FeatureToCopy          = {polygon05,polygon06,polygon07,polygon08,polygon09,p
olygon10,polygon11,polygon12,polygon13,polygon14,polygon15,polygon16};
      CopyFrom               = point02;
      CopyTo                 = point06;
   }
}

transaction modelBased "Graph changed by user"
{
   feature point47 GC.Point
   {
      XTranslation           = <free> (51.3183059913976);
      ZTranslation           = <free> (107.573080397562);
   }
   feature point53 GC.Point
   {
      XTranslation           = <free> (21.1112275003648);
      ZTranslation           = <free> (136.713573825957);
   }
   feature point59 GC.Point
   {
      XTranslation           = <free> (22.4099326546715);
      ZTranslation           = <free> (182.23441546802);
   }
}

transaction modelBased "Graph changed by user"
{
   feature floor GC.GraphVariable
   {
      Value                  = 96.0;
   }
}

transaction modelBased "Graph changed by user"
{
   feature coordinateSystem22 GC.CoordinateSystem
   {
      Origin                 = point66;
```

```
   PrimaryDirection        = direction08;
   PrimaryAxis             = AxisOption.Z;
   SecondaryDirection      = baseCS.XDirection;
   SecondaryAxis           = AxisOption.X;
   Visible                 = false;
}
feature coordinateSystem23 GC.CoordinateSystem
{
   CoordinateSystem        = coordinateSystem22;
   XTranslation            = 0;
   YTranslation            = 0;
   ZTranslation            = 3;
   XRotation               = 180;
   YRotation               = 0;
   ZRotation               = 180;
}
feature direction08 GC.Direction
{
   Origin                  = point59;
   DirectionPoint          = point65;
}
feature line28 GC.Line
{
   StartPoint              = point59;
   EndPoint                = point65;
}
feature line29 GC.Line
{
   StartPoint              = point67;
   Direction               = coordinateSystem23.ZDirection;
   Length                  = 1;
}
feature line30 GC.Line
{
   StartPoint              = point68;
   Direction               = coordinateSystem23.ZDirection;
   Length                  = 1;
}
feature point59 GC.Point
{
   XTranslation            = <free> (28.0359380459788);
}
feature point65 GC.Point
{
   CoordinateSystem        = coordinateSystem01;
   XTranslation            = <free> (22.4099326546715);
```

```
    YTranslation            = 0;
    ZTranslation            = <free> (220);
  }
  feature point66 GC.Point
  {
    Curve                   = line28;
    T                       = <free> (0.0942357304092773);
    HandlesVisible          = true;
    Visible                 = false;
  }
  feature point67 GC.Point
  {
    FeatureToCopy           = point61;
    CopyFrom                = coordinateSystem21;
    CopyTo                  = coordinateSystem23;
  }
  feature point68 GC.Point
  {
    FeatureToCopy           = point62;
    CopyFrom                = coordinateSystem21;
    CopyTo                  = coordinateSystem23;
  }
  feature point69 GC.Point
  {
    Curve0                  = line30;
    Curve1                  = line26;
  }
  feature point70 GC.Point
  {
    Curve0                  = line29;
    Curve1                  = line27;
  }
  feature polygon22 GC.Polygon
  {
    Vertices                = {point53,point64,point69,point59};
  }
  feature polygon23 GC.Polygon
  {
    Vertices                = {point53,point63,point70,point59};
  }
}

transaction modelBased "Graph changed by user"
{
  feature direction09 GC.Direction
  {
```

```
    Origin                  = point65;
    DirectionPoint          = point71;
  }
  feature line31 GC.Line
  {
    StartPoint              = point71;
    EndPoint                = point65;
  }
  feature point65 GC.Point
  {
    ZTranslation            = 220;
  }
  feature point71 GC.Point
  {
    CoordinateSystem        = coordinateSystem01;
    XTranslation            = <free> (22.4099326546715);
    YTranslation            = 0;
    ZTranslation            = 260;
  }
}

transaction modelBased "Graph changed by user"
{
  feature coordinateSystem24 GC.CoordinateSystem
  {
    Origin                  = point72;
    PrimaryDirection        = direction09;
    PrimaryAxis             = AxisOption.Z;
    SecondaryDirection      = baseCS.XDirection;
    SecondaryAxis           = AxisOption.X;
  }
  feature coordinateSystem25 GC.CoordinateSystem
  {
    CoordinateSystem        = coordinateSystem24;
    XTranslation            = 0;
    YTranslation            = 0;
    ZTranslation            = 3;
    XRotation               = 0;
    YRotation               = 0;
    ZRotation               = 0;
  }
  feature point72 GC.Point
  {
    Curve                   = line31;
    T                       = <free> (0.852036366873337);
    HandlesVisible          = true;
```

```
    }
}

transaction modelBased “Graph changed by user”
{
    feature coordinateSystem24 GC.CoordinateSystem
    {
        Visible                 = false;
    }
    feature coordinateSystem25 GC.CoordinateSystem
    {
        ZTranslation             = -3;
        XRotation                = 180;
        ZRotation                = 180;
    }
    feature coordinateSystem26 GC.CoordinateSystem
    {
        Origin                  = point78;
        PrimaryDirection          = direction10;
        PrimaryAxis              = AxisOption.Z;
        SecondaryDirection        = baseCS.XDirection;
        SecondaryAxis             = AxisOption.X;
        Visible                 = false;
    }
    feature coordinateSystem27 GC.CoordinateSystem
    {
        CoordinateSystem          = coordinateSystem26;
        XTranslation             = 0;
        YTranslation             = 0;
        ZTranslation             = 3;
        XRotation                = 0;
        YRotation                = 0;
        ZRotation                = 180;
        Visible                 = false;
    }
    feature direction09 GC.Direction
    {
        Origin                  = point71;
        DirectionPoint           = point65;
    }
    feature direction10 GC.Direction
    {
        Origin                  = point77;
        DirectionPoint           = point71;
    }
    feature line32 GC.Line
```

```
{
   StartPoint              = point73;
   Direction               = coordinateSystem25.ZDirection;
   Length                  = 1;
}
feature line33 GC.Line
{
   StartPoint              = point74;
   Direction               = coordinateSystem25.ZDirection;
   Length                  = 1;
}
feature line34 GC.Line
{
   StartPoint              = point77;
   EndPoint                = point71;
}
feature point72 GC.Point
{
   Visible                 = false;
}
feature point73 GC.Point
{
   FeatureToCopy           = point68;
   CopyFrom                = coordinateSystem23;
   CopyTo                  = coordinateSystem25;
}
feature point74 GC.Point
{
   FeatureToCopy           = point67;
   CopyFrom                = coordinateSystem23;
   CopyTo                  = coordinateSystem25;
}
feature point75 GC.Point
{
   Curve0                  = line30;
   Curve1                  = line32;
}
feature point76 GC.Point
{
   Curve0                  = line29;
   Curve1                  = line33;
}
feature point77 GC.Point
{
   CoordinateSystem        = coordinateSystem01;
   XTranslation            = <free> (22.4099326546715);
```

```
   YTranslation            = 0;
   ZTranslation            = 310;
}
feature point78 GC.Point
{
   Curve                   = line34;
   T                       = <free> (0.122508582782041);
   HandlesVisible          = true;
   Visible                 = false;
}
feature point79 GC.Point
{
   FeatureToCopy           = point74;
   CopyFrom                = coordinateSystem25;
   CopyTo                  = coordinateSystem27;
   Visible                 = false;
}
feature point80 GC.Point
{
   FeatureToCopy           = point73;
   CopyFrom                = coordinateSystem25;
   CopyTo                  = coordinateSystem27;
   Visible                 = false;
}
feature point81 GC.Point
{
   FeatureToCopy           = point80;
   CopyFrom                = coordinateSystem27;
   CopyTo                  = point77;
}
feature point82 GC.Point
{
   FeatureToCopy           = point79;
   CopyFrom                = coordinateSystem27;
   CopyTo                  = point77;
}
feature polygon24 GC.Polygon
{
   Vertices                = {point59,point65,point75,point69};
}
feature polygon25 GC.Polygon
{
   Vertices                = {point59,point70,point76,point65};
}
feature polygon26 GC.Polygon
{
```

```
      Vertices              = {point65,point77,point82,point76};
   }
   feature polygon27 GC.Polygon
   {
      Vertices              = {point65,point77,point81,point75};
   }
}

transaction modelBased “Graph changed by user”
{
   feature line32 GC.Line
   {
      SymbolXY               = {101, 210};
   }
   feature point47 GC.Point
   {
      XTranslation           = <free> (45.6779803257499);
   }
   feature polygon28 GC.Polygon
   {
      FeatureToCopy          = {polygon22,polygon23,polygon24,polygon25,polygon26,p
olygon27};
      CopyFrom               = point02;
      CopyTo                 = point06;
   }
   feature polygon29 GC.Polygon
   {
      FeatureToCopy          = {polygon22,polygon23,polygon24,polygon25,polygon26,p
olygon27};
      CopyFrom               = point02;
      CopyTo                 = point04;
   }
   feature polygon30 GC.Polygon
   {
      FeatureToCopy          = {polygon22,polygon23,polygon24,polygon25,polygon26,p
olygon27};
      CopyFrom               = point02;
      CopyTo                 = point05;
   }
}

transaction modelBased “Graph changed by user”
{

   feature point35 GC.Point
   {
```

```
      XTranslation            = <free> (55.149521025395);
      ZTranslation            = <free> (51.7275713927391);
   }
   feature point41 GC.Point
   {
      ZTranslation            = <free> (82.2517836115805);
   }
   feature point47 GC.Point
   {
      XTranslation            = <free> (11.4699727189354);
   }
   feature point53 GC.Point
   {
      XTranslation            = <free> (-13.8145319813549);
      ZTranslation            = <free> (138.470352358222);
   }
   feature point59 GC.Point
   {
      XTranslation            = <free> (-86.0358017195482);
      ZTranslation            = <free> (165.218068962452);
   }
   feature point65 GC.Point
   {
      XTranslation            = <free> (-130.291150930839);
      ZTranslation            = <free> (142.698846844569);
   }
}

transaction modelBased “Graph changed by user”
{
   feature point53 GC.Point
   {
      ZTranslation            = <free> (153.553133331525);
   }
   feature polygon31 GC.Polygon
   {
      FeatureToCopy             = {polygon22,polygon23,polygon24,polygon25};
      CopyFrom                  = point02;
      CopyTo                  = point04;
   }
   feature polygon32 GC.Polygon
   {
      FeatureToCopy             = {polygon22,polygon23,polygon24,polygon25};
      CopyFrom                  = point02;
      CopyTo                  = point05;
   }
```

```
    feature polygon33 GC.Polygon
    {
      FeatureToCopy          = {polygon22,polygon23,polygon24,polygon25};
      CopyFrom               = point02;
      CopyTo                 = point06;
    }
}

transaction modelBased “Graph changed by user”
{
    feature direction09 GC.Direction
    {
      Visible                = false;
    }
    feature line31 GC.Line
    {
      Visible                = false;
    }
    feature point71 GC.Point
    {
      Visible                = false;
    }
    feature polygon32 GC.Polygon
    {
      SymbolXY               = {107, 217};
    }
}

transaction modelBased “Graph changed by user”
{
    feature coordinateSystem25 GC.CoordinateSystem
    {
      Visible                = false;
    }
    feature line29 GC.Line
    {
      Visible                = false;
    }
    feature line30 GC.Line
    {
      Visible                = false;
    }
    feature line32 GC.Line
    {
      Visible                = false;
    }
```

```
feature line33 GC.Line
{
   Visible                 = false;
}
feature point03 GC.Point
{
   Visible                 = false;
}
feature point04 GC.Point
{
   Visible                 = false;
}
feature point05 GC.Point
{
   Visible                 = false;
}
feature point06 GC.Point
{
   Visible                 = false;
}
feature point08 GC.Point
{
   Visible                 = false;
}
feature point59 GC.Point
{
   XTranslation            = <free> (-100.934478078732);
}
feature point61 GC.Point
{
   Visible                 = false;
}
feature point62 GC.Point
{
   Visible                 = false;
}
feature point65 GC.Point
{
   ZTranslation            = <free> (87.5748570790225);
}
feature point67 GC.Point
{
   Visible                 = false;
}
feature point68 GC.Point
{
```

```
        Visible                 = false;
    }
    feature point73 GC.Point
    {
        Visible                 = false;
    }
    feature point74 GC.Point
    {
        Visible                 = false;
    }
}
```

REFERENCES

ARCspace.com. < http://www.arcspace.com/architects/foreign_office/yokohama/yokohama_index.htm>.

Al-Haddad, T. (2006). PerFORMance: Integrating Structural Feedback into Design Processes for Complex Surface-Active Form. College of Architecture. Atlanta, Georgia Institute of Technology. Masters of Architecture: 127.

Aranda, B. and C. Lasch, Eds. (2006). Tooling. Princeton Architectural Press, New York.

Hensel, M. and A. Menges, Eds. (2006). Morpho-Ecologies, AA Publications.

Hensel, M., A. Menges, et al., Eds. (2004). Emergence: Morphogenetic Design Strategies.

Hoberman, C. (2004). “Unfolding Architecture.” Architectural Design: 3.

Kolarevic, B., Ed. (2003). Architecture in the Digital Age: Design and Manufacturing, Spon Press.

Lynn, G., C. Hoberman, et al., Eds. (2004). Folding in Architecture, Wiley Academy.

McQuaid, Matilda. Shigeru Ban. New York: Phaidon, 2003

Reeser, A. and A. Schafer, Eds. (2004). Praxis: New Technologies://New Architectures. Praxis.

Spuybroek, L. (2004). NOX Machining Architecture, Thames & Hudson.

Vyzoviti, S. (2003). Folding Architecture: Spatial, Structural and Organizational Diagrams.